RECEIVED
SEP 24 5 55 PM '01
POSTAL RATE COMMISSION
OFFICE OF THE SECRETARY

Docket No. R2001-1

USPS-LR-J-58

First-Class, Periodicals, and Standard Mail Weight Studies

Table of Contents

List of Tables

I. Introduction

Four analyses are presented in this library reference. This library reference provides the supporting documentation and analyses used to estimate test year volume-variable costs by weight increment. A test year parcel/flat cost difference is also calculated for Standard mail. In addition, the test year cost difference due to differences in presorting and drop shipment of Standard mail is calculated. Finally, volume distributions by destination entry and weight increment are developed for Standard ECR mail.

This is a Category 2 library reference sponsored by witness Schenk (USPS-T-43). The costs by weight increment analyses in this library reference update previous studies (USPS-LR-I-91, USPS-LR-I-92, and USPS-LR-I-93) sponsored by USPS witness Daniel (USPS-T-28/R2000-1). The test year parcel/flat cost difference and test year cost difference due to differences in presorting and dropshipment updates previous analysis by witness Crum (USPS-T-27/R2000-1) as reported in Attachment F, Tables 4 and Figure 2, respectively. In all cases, the same methodology is used in this library reference as was used in the previous studies, with the cost and volume data updated.

Other testimony and library references referred to in this library reference include:

- USPS-T-11 for BY00 CRA costs
- USPS-T-13 for volume-variable cost methodology
- USPS LR-J-10 for the IOCS data set
- USPS LR-J-112 for volumes by shape and weight increment

Witnesses Robinson (USPS-T-29), Hope (USPS-T-31), Moeller (USPS-T-32), and Taufique (USPS-T-34) use the cost by weight increment estimates in this library reference as a general reference in developing First-Class, Standard ECR, Standard Regular, and Periodicals rate designs. Witness Mayes reports the test year parcel/flat cost differential for Standard Mail in USPS-T-23. Witness Moeller (USPS-T-32) uses the test year parcel/flat cost differential and the test year cost difference due to differences in presorting and drop shipment of Standard mail in developing the Standard Regular rate design. Witness Hope (USPS-T-31) uses the volume distribution by destination entry and weight increment for Standard ECR mail in developing the rate design for that mail class.

II. Organization

Four sets of analyses are presented in this library reference: costs by weight increment estimates by class, an estimate of the test year Standard parcel/flat cost differential, an estimate of the test year cost difference due to differences in presorting and drop shipment of Standard mail, and the volume distributions by destination entry and weight increment.

The cost by weight increment estimates are provided in the following Excel workbooks:

- **LR58ASP.xls** – Develops the test year First-Class single piece unit costs by shape by weight increment. Final results are reported in Table 1.
- **LR58PRE.xls** – Develops the test year First-Class presort unit costs by shape by weight increment. Final results are reported in Table 2.
- **LR58PER.xls** – Develops the test year Periodicals unit costs by shape by weight increment. Final results are reported in Table 3.

- **LR58AREG.xls** – Develops the test year Standard Regular rate unit costs by shape by weight increment. Final results are reported in Table 4.
- **LR58AECR.xls** – Develops the test year Standard ECR unit costs by shape by weight increment. Final results are reported in Table 4.

The underlying mail processing and window service costs by weight increment for clerks and mailhandlers is calculated using a similar methodology to that used in USPS-LR-I-99/R2000-1, sponsored by witness Daniel. Distribution keys are developed using the 2000 In-Office Cost System (IOCS) data set and RPW volumes. The current study uses FY2000 IOCS data and the Postal Service's cost distribution methodology. City Carrier In-Office costs by weight increment are estimated using a similar methodology to that described in USPS-LR-I-100/R2000-1, sponsored by witness Daniel. The only difference is that the current study uses FY2000 IOCS data.

The parcel/flat cost differential analysis is presented in Table 5 in Excel workbook 'LR58STDCBS.xls.' This spreadsheet also reports the test year Standard Regular Rate and ECR unit costs by shape by cost segment in the 'Data' spreadsheet.

The test year cost difference due to differences in presorting and drop shipment for Standard mail is presented in Excel workbook 'LR58ADJ.xls' in Tables 6 and in sheet 'Summary.'

Volume distributions by destination entry and weight increment are developed for Standard ECR mail using volume data from USPS-LR-J-112. The volume distributions are given in the Excel workbook 'tiers_table.xls.' Witness Hope (USPS-T-31) uses these estimates in developing the rate design for Standard ECR mail. These volume distributions are presented in Table 7.

Table 1:
Costs by Ounce Increment for First-Class Single-Piece

	0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13+	Total
volume	41,198,031,508	3,206,024,638	1,237,571,918	688,326,214	443,037,984	301,176,958	220,957,935	168,664,575	136,575,010	102,796,992	82,294,754	65,168,037	48,762,520	47,899,389,025
pounds	1,074,310,424	290,303,921	192,185,342	150,553,618	124,943,760	103,875,077	90,179,758	79,483,329	72,846,356	61,291,141	54,220,431	46,958,291	38,201,350	2,379,332,797
cubic feet (weight/density)	48,220,459	15,635,390	11,807,544	9,868,083	8,619,415	7,344,416	6,622,474	5,922,376	5,861,068	4,816,443	4,322,288	3,823,037	3,171,182	135,834,152
all mp (3.1) tally	5,007,647	1,050,902	452,531	380,700	138,433	91,055	68,637	69,959	44,862	48,899	38,910	36,503	22,883	7,431,921
window service (3.2) tally	612,630	43,508	15,753	13,657	8,121	5,260	5,015	3,775	2,514	3,165	1,385	837	826	716,445
delivery in-office (6.1) tally	1,218,481	161,416	60,954	42,037	13,861	9,102	7,036	5,995	4,031	2,836	3,321	1,715	1,078	1,531,861
delivery in-office (6.2) 6.1	232,209	30,761	11,616	8,011	2,641	1,735	1,341	1,142	768	540	633	327	205	291,931
del. route (7.1) piece	9,810	763	295	164	105	72	53	40	33	24	20	16	12	11,406
del. access (7.2) piece	120,864	9,406	3,631	2,019	1,300	884	648	495	401	302	241	191	143	140,525
elem. load (7.3)shape&pc	254,209	24,733	13,410	9,007	7,117	5,312	4,546	3,651	3,446	2,691	2,263	1,918	1,521	333,824
del. support (7.4) sum6&7	291,001	35,631	14,209	9,660	4,062	2,789	2,229	1,848	1,436	1,062	1,061	697	500	386,185
vehicle service (8) cube	17,607	5,709	4,311	3,603	3,147	2,682	2,418	2,162	2,067	1,759	1,578	1,396	1,158	49,597
delivery rural (10)shape&pc	321,160	32,478	14,667	8,660	5,717	3,930	2,906	2,229	1,815	1,369	1,097	872	654	397,554
air/water trans. (14) weight	128,977	34,853	23,073	18,075	15,000	12,471	10,827	9,540	8,746	7,358	6,509	5,638	4,586	285,852
hwy/rail trans. (14)cube	112,690	36,540	27,594	23,061	20,143	17,164	15,477	13,840	13,230	11,256	10,101	8,934	7,411	317,441
Other weight	196,659	53,742	35,161	27,560	22,672	19,016	14,609	14,548	13,335	11,220	9,925	8,566	6,993	436,551
Total Cost	8,525,644	1,519,842	677,224	526,214	242,530	171,470	137,639	129,223	96,662	[illegible]	77,045	67,636	47,971	12,308,093
Total Unit Cost	$ 0.207	$ 0.474	$ 0.547	$ 0.764	$ 0.547	$ 0.569	$ 0.623	$ 0.766	$ 0.708	$ 0.900	$ 0.936	$ 1.038	$ 0.984	$ 0.257
number of additional ounces purchased		1	2	3	4	5	6	7	8	9	10	11	12	
total number of additional ounces purchased		3,206,024,638	2,475,143,837	2,064,978,642	1,772,151,856	1,505,884,792	1,325,747,613	1,180,652,024	1,092,600,061	925,172,928	822,947,544	716,848,410	585,150,243	17,673,302,608
cost of pieces in excess of first ounce cost		$ 858,509,768	$ 421,168,471	$ 383,796,121	$ 150,654,617	$ 109,156,277	$ 91,922,855	$ 94,328,017	$ 68,424,644	$ 71,211,681	$ 60,017,786	$ 54,154,565	$ 37,861,662	$ 2,399,426,960
Marginal Cost Difference		$ 0.267	$ 0.073	$ 0.217	$ (0.217)	$ 0.022	$ 0.054	$ 0.143	$ (0.058)	$ 0.192	$ 0.037	$ 0.102	$ (0.064)	

Table 1:
Costs by Ounce Increment for First-Class Single-Piece

	0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13+	Total
volume	41,198,031,508	3,206,024,638	1,237,571,918	688,326,214	443,037,964	301,176,958	220,957,935	168,664,575	136,575,010	102,796,992	82,294,754	65,168,037	48,762,520	47,899,389,025
pounds	1,074,310,424	290,303,921	192,185,342	150,553,618	124,943,760	103,875,077	90,179,758	79,463,329	72,846,356	61,291,141	54,220,431	46,958,291	38,201,350	2,379,332,797
cubic feet (weight/density)	48,220,459	15,635,390	11,807,544	9,868,083	8,619,415	7,344,416	6,622,474	5,922,376	5,661,068	4,816,443	4,322,266	3,823,037	3,171,182	135,834,152
all mp (3.1) tally	5,007,647	1,050,902	452,531	360,700	138,433	91,055	68,637	89,959	44,862	48,899	38,910	36,503	22,883	7,431,921
window service (3.2) tally	612,630	43,508	15,753	13,657	8,121	5,260	5,015	3,775	2,514	3,165	1,385	837	826	716,445
delivery in-office (6.1) tally	1,218,481	161,416	60,954	42,037	13,861	9,102	7,036	5,995	4,031	2,836	3,321	1,715	1,078	1,531,861
delivery in-office (6.2) 6.1	232,209	30,761	11,616	8,011	2,641	1,735	1,341	1,142	768	540	633	327	205	291,931
del. route (7.1) piece	9,810	763	295	164	105	72	53	40	33	24	20	16	12	11,406
del. access (7.2) piece	120,864	9,406	3,631	2,019	1,300	884	648	495	401	302	241	191	143	140,525
elem. load (7.3)shape&pc	254,209	24,733	13,410	9,007	7,117	5,312	4,546	3,651	3,446	2,691	2,263	1,918	1,521	333,824
del. support (7.4) sum6&7	291,001	35,631	14,209	9,660	4,062	2,789	2,229	1,848	1,436	1,062	1,061	697	500	366,185
vehicle service (8) cube	17,607	5,709	4,311	3,603	3,147	2,682	2,418	2,162	2,067	1,759	1,578	1,396	1,158	49,597
delivery rural (10)shape&pc	321,160	32,478	14,667	8,660	5,717	3,930	2,906	2,229	1,815	1,369	1,097	872	654	397,554
air/water trans. (14) weight	128,977	34,853	23,073	18,075	15,000	12,471	10,827	9,540	8,746	7,358	6,509	5,638	4,586	285,652
hwy/rail trans. (14)cube	112,690	36,540	27,594	23,061	20,143	17,164	15,477	13,840	13,230	11,256	10,101	8,934	7,411	317,441
Other weight	198,659	53,142	35,181	27,560	22,872	19,015	16,568	14,546	13,335	11,220	9,925	8,096	6,903	435,551
Total Cost	$ 8,523,944	$ 1,519,842	$ 677,224	$ 526,214	$ 242,520	$ 171,470	$ 137,639	$ 129,223	$ 98,652	$ 92,481	$ 77,067	$ 67,056	$ 47,971	$ 12,308,893
Total Unit Cost	$ 0.207	$ 0.474	$ 0.547	$ 0.764	$ 0.547	$ 0.569	$ 0.623	$ 0.766	$ 0.708	$ 0.900	$ 0.936	$ 1.029	$ 0.984	$ 0.257
		1	2	3	4	5	6	7	8	9	10	11	12	
number of additional ounces purchased		3,206,024,638	2,475,143,837	2,064,978,642	1,772,151,856	1,505,884,792	1,325,747,613	1,180,652,024	1,092,600,081	925,172,928	822,947,544	716,848,410	585,150,243	17,673,302,608
total number of additional ounces purchased		$ 656,509,768	$ 421,168,471	$ 383,798,121	$ 180,884,617	$ 109,156,277	$ 91,922,855	$ 94,326,017	$ 68,424,844	$ 71,211,681	$ 60,017,768	$ 54,154,689	$ 37,881,652	$ 2,369,426,960
cost of pieces in excess of first ounce cost														
Marginal Cost Difference		$ 0.267	$ 0.073	$ 0.217	$ (0.217)	$ 0.022	$ 0.054	$ 0.143	$ (0.058)	$ 0.192	$ 0.037	$ 0.102	$ (0.054)	

Table 1:
Costs by Ounce Increment for First-Class Single-Piece

	0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13+	Total
volume	41,198,031,508	3,206,024,638	1,237,571,918	688,326,214	443,037,964	301,176,858	220,957,935	168,664,575	136,575,010	102,796,992	82,294,754	65,168,037	48,762,520	47,899,389,025
pounds	1,074,310,424	290,303,921	192,185,342	150,553,618	124,943,760	103,875,077	90,179,758	79,463,329	72,846,356	61,291,141	54,220,431	46,958,291	38,201,350	2,379,332,797
cubic feet (weight/density)	48,220,459	15,635,390	11,807,544	9,868,083	8,819,415	7,344,416	6,622,474	5,922,376	5,661,068	4,816,443	4,322,266	3,823,037	3,171,182	135,834,152
all mp (3.1) tally	5,007,647	1,050,902	452,531	360,700	138,433	91,055	68,637	89,959	44,882	48,899	38,910	36,503	22,883	7,431,921
window service (3.2) tally	612,630	43,508	15,753	13,657	8,121	5,260	5,015	3,775	2,514	3,165	1,385	837	826	716,445
delivery in-office (6.1) tally	1,218,481	161,416	60,954	42,037	13,861	9,102	7,036	5,995	4,031	2,836	3,321	1,715	1,078	1,531,861
delivery in-office (6.2) 6.1	232,209	30,761	11,616	8,011	2,641	1,735	1,341	1,142	768	540	633	327	205	291,931
del. route (7.1) piece	8,810	783	295	164	105	72	53	40	33	24	20	16	12	11,406
del. access (7.2) piece	120,864	9,406	3,631	2,019	1,300	884	648	495	401	302	241	191	143	140,525
elem. load (7.3)shape&wt	215,619	36,842	16,159	10,322	8,572	7,222	6,908	6,229	6,460	5,575	5,164	4,713	4,041	333,824
del. support (7.4) sum6&7	283,997	37,628	14,708	9,899	4,326	3,136	2,658	2,316	1,983	1,586	1,587	1,204	957	366,185
vehicle service (8) cube	17,607	5,709	4,311	3,603	3,147	2,882	2,418	2,162	2,067	1,759	1,578	1,396	1,158	49,597
delivery rural (10)shape&pc	321,160	32,478	14,667	8,660	5,717	3,930	2,906	2,229	1,815	1,369	1,097	872	654	397,554
air/water trans. (14) weight	128,977	34,853	23,073	18,075	15,000	12,471	10,827	9,540	8,746	7,358	6,509	5,638	4,586	285,652
hwy/rail trans. (14)cube	112,890	36,540	27,594	23,061	20,143	17,164	15,477	13,840	13,230	11,256	10,101	8,934	7,411	317,441
Other weight	199,938	54,028	35,767	28,019	23,253	19,332	16,783	14,789	13,557	11,407	10,091	8,739	7,110	442,814
Total Cost	8,481,628	1,535,034	681,058	528,227	244,620	174,043	140,706	132,512	100,466	96,074	80,638	71,085	51,064	12,317,156
Total Unit Cost	$ 0.206	$ 0.479	$ 0.550	$ 0.767	$ 0.552	$ 0.578	$ 0.637	$ 0.786	$ 0.736	$ 0.935	$ 0.980	$ 1.091	$ 1.047	$ 0.257
number of additional ounces purchased		1	2	3	4	5	6	7	8	9	10	11	12	
total number of additional ounces purchased		3,206,024,638	2,475,143,837	2,064,978,642	1,772,151,856	1,505,884,792	1,325,747,613	1,180,652,024	1,092,600,081	925,172,928	822,947,544	716,848,410	585,150,243	17,673,302,608
cost of pieces in excess of first ounce cost	$ 874,994,825	$ 426,273,469	$ 386,517,952	$ 153,409,896	$ 112,038,637	$ 95,216,261	$ 97,788,021	$ 72,348,782	$ 74,911,146	$ 63,696,098	$ 57,668,087	$ 41,025,196	$ 2,455,888,369	0.1390
Marginal Cost Difference	$ 0.273	$ 0.072	$ 0.217	$ (0.215)	$ 0.026	$ 0.059	$ 0.149	$ (0.050)	$ 0.199	$ 0.045	$ 0.111	$ (0.044)		

Table 2:
Costs by Ounce Increment for First-Class Presort

	0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13+	Total
volume	49,265,935,104	1,471,054,864	312,572,529	107,693,132	46,151,530	22,219,361	14,069,551	16,436,932	12,545,955	10,156,263	8,676,355	7,595,785	4,105,715	51,239,696,072
pounds	1,718,048,677	122,224,037	47,051,977	22,468,871	12,228,995	7,110,467	5,329,363	7,230,596	6,194,728	5,666,307	5,364,195	5,162,501	2,997,745	1,934,482,387
cubic feet (weight/density)	70,891,552	5,385,230	2,344,815	1,182,565	660,958	398,575	299,562	407,281	344,407	316,841	298,754	286,802	165,880	81,163,257
all mp (3.1) tally	2,191,161	240,812	60,246	40,529	14,382	5,921	4,800	6,322	1,708	1,196	1,002	1,553	698	2,557,850
window service (3.2) tally	46,148	2,310	721	476	31	15	9	11	8	7	6	5	3	49,711
delivery in-office (6.1) tally	788,830	61,483	15,214	10,076	1,534	1,659	1,188	1,409	225	830	106	399	77	859,983
delivery in-office (6.2) 6.1	144,373	11,545	2,857	1,892	288	312	223	265	42	156	20	75	14	161,490
del. route (7.1) piece	12,416	371	79	27	12	6	4	4	3	3	2	2	1	12,916
del. access (7.2) piece	75,096	2,242	476	164	70	34	21	25	19	15	13	12	6	78,105
elem. load (7.3)shape&wt	372,289	22,696	6,159	2,254	1,187	666	499	664	438	442	387	343	173	405,750
del. support (7.4) sum6&7	218,319	15,422	3,896	2,225	498	420	304	373	122	229	90	134	46	241,085
vehicle service (8) cube	33,990	2,582	1,124	567	317	191	144	195	165	152	143	138	80	38,914
delivery rural (10)shape&pc	401,696	11,747	2,407	812	351	172	109	127	92	76	64	56	29	417,295
air/water trans. (14) weight	193,077	13,736	5,288	2,525	1,374	799	599	813	696	637	603	580	337	217,398
hwy/rail trans. (14)cube	149,099	11,326	4,932	2,487	1,390	838	630	857	724	666	628	603	349	170,702
Other weight	116,560	8,292	3,192	1,524	830	482	362	491	420	384	364	350	203	131,243
Total Cost	4,723,067	404,665	105,591	65,669	22,263	11,515	8,891	11,555	4,664	4,792	3,429	4,260	2,016	5,342,441
Total Unit Cost	$ 0.096	$ 0.275	$ 0.338	$ 0.609	$ 0.482	$ 0.518	$ 0.632	$ 0.703	$ 0.372	$ 0.472	$ 0.395	$ 0.560	$ 0.491	$ 0.104
number of additional ounces purchased		1	2	3	4	5	6	7	8	9	10	10	10	
total number of additional ounces purchased		1,471,054,864	625,145,058	323,079,397	184,606,120	111,096,807	84,417,307	115,058,523	100,367,640	91,406,366	86,763,555	75,957,847	41,057,146	3,310,010,630
cost of pieces in excess of first ounce cost		$ 263,536,723	$ 76,625,266	$ 55,234,975	$ 17,836,696	$ 9,664,561	$ 7,542,503	$ 9,979,143	$ 3,460,987	$ 3,616,713	$ 2,597,119	$ 3,522,076	$ 1,622,372	$ 465,152,836
Marginal Cost Difference		$ 0.179	$ 0.066	$ 0.268	$ (0.126)	$ 0.036	$ 0.114	$ 0.071	$ (0.331)	$ 0.100	$ (0.077)	$ 0.164	$ (0.069)	0.1376

Table 2:
Costs by Ounce Increment for First-Class Presort

	0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13+	Total
volume	49,265,935,104	1,471,054,864	312,572,529	107,693,132	46,151,530	22,219,361	14,069,551	16,436,932	12,545,955	10,156,263	8,676,355	7,595,785	4,105,715	51,239,696,072
pounds	1,718,048,677	122,224,037	47,051,977	22,468,871	12,228,995	7,110,467	5,329,363	7,230,596	6,194,728	5,666,307	5,364,195	5,162,501	2,997,745	1,934,462,387
cubic feet (weight/density)	70,891,552	5,385,230	2,344,815	1,182,565	660,958	398,575	299,562	407,281	344,407	316,841	298,754	286,802	165,880	81,163,257
all mp (3.1) tally	2,191,161	240,812	60,246	40,529	14,382	5,921	4,800	6,322	1,708	1,196	1,002	1,553	698	2,557,850
window service (3.2) tally	46,148	2,310	721	476	31	15	9	11	8	7	6	5	3	49,711
delivery in-office (6.1) tally	768,830	61,483	15,214	10,076	1,534	1,659	1,188	1,409	225	830	106	399	77	859,983
delivery in-office (6.2) 6.1	144,373	11,545	2,857	1,892	288	312	223	265	42	156	20	75	14	161,490
del. route (7.1) piece	12,418	371	79	27	12	6	4	4	3	3	2	2	1	12,916
del. access (7.2) piece	75,096	2,242	476	164	70	34	21	25	19	15	13	12	6	78,105
elem. load (7.3)shape&wt	372,289	22,696	6,159	2,254	1,187	666	499	664	438	442	387	343	173	405,750
del. support (7.4) sum6&7	218,319	15,422	3,896	2,225	498	420	304	373	122	229	90	134	46	241,085
vehicle service (8) cube	33,990	2,582	1,124	567	317	191	144	195	165	152	143	138	80	38,914
delivery rural (10)shape&pc	401,696	11,747	2,407	812	351	172	109	127	92	76	64	56	29	417,295
air/water trans. (14) weight	193,077	13,736	5,288	2,525	1,374	799	599	813	696	637	603	580	337	217,398
hwy/rail trans. (14)cube	149,099	11,326	4,932	2,487	1,390	838	630	857	724	666	628	603	349	170,702
Other weight	116,560	8,292	3,192	1,524	830	482	362	491	420	384	364	350	203	131,243
Total Cost	4,723,057	404,565	106,591	65,559	22,263	11,515	8,891	11,555	4,664	4,792	3,429	4,250	2,016	5,342,441
Total Unit Cost	$ 0.096	$ 0.275	$ 0.341	$ 0.609	$ 0.482	$ 0.518	$ 0.632	$ 0.703	$ 0.372	$ 0.472	$ 0.395	$ 0.560	$ 0.491	$ 0.104
number of additional ounces purchased		1	2	3	4	5	6	7	8	9	10	10	10	
total number of additional ounces purchased		1,471,054,864	625,145,058	323,079,397	184,606,120	111,096,807	84,417,307	115,058,523	100,367,640	91,406,366	86,763,555	75,957,847	41,057,146	3,310,010,630
cost of pieces in excess of first ounce cost		$ 263,536,723	$ 76,625,266	$ 55,234,275	$ 17,838,896	$ 9,384,581	$ 7,542,503	$ 9,979,143	$ 3,460,967	$ 3,818,713	$ 2,597,119	$ 3,522,075	$ 1,622,374	$ 455,162,636
														0.1375
Marginal Cost Difference		$ 0.179	$ 0.066	$ 0.268	$ (0.126)	$ 0.036	$ 0.114	$ 0.071	$ (0.331)	$ 0.100	$ (0.077)	$ 0.164	$ (0.069)	

Table 2:
Costs by Ounce Increment for First-Class Presort

	0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13+	Total
volume	49,265,935,104	1,471,054,864	312,572,529	107,693,132	46,151,530	22,219,361	14,069,551	16,436,932	12,545,955	10,156,263	8,676,355	7,595,785	4,105,715	51,239,696,072
pounds	1,718,048,677	122,224,037	47,051,977	22,468,871	12,228,995	7,110,467	5,329,383	7,230,596	6,194,728	5,666,307	5,364,195	5,162,501	2,997,745	1,934,462,387
cubic feet (weight/density)	70,891,552	5,385,230	2,344,815	1,182,565	660,958	398,575	299,562	407,261	344,407	316,841	298,754	286,802	165,880	81,163,257
all mp (3.1) tally	2,191,161	240,812	60,246	40,529	14,382	5,921	4,800	6,322	1,708	1,196	1,002	1,553	698	2,557,850
window service (3.2) tally	46,148	2,310	721	476	31	15	9	11	8	7	6	5	3	49,711
delivery in-office (6.1) tally	768,830	61,483	15,214	10,076	1,534	1,659	1,188	1,409	225	830	106	399	77	859,983
delivery in-office (6.2) 6.1	144,373	11,545	2,857	1,892	288	312	223	265	42	156	20	75	14	161,490
del. route (7.1) piece	12,118	371	79	27	12	6	4	4	3	3	2	2	1	12,916
del. access (7.2) piece	75,096	2,242	476	164	70	34	21	25	19	15	13	12	6	78,105
elem. load (7.3)shape&wt	372,289	22,696	6,159	2,254	1,187	666	499	664	438	442	387	343	173	405,750
del. support (7.4) sum6&7	218,319	15,422	3,896	2,225	498	420	304	373	122	229	90	134	46	241,085
vehicle service (8) cube	33,990	2,582	1,124	567	317	191	144	195	165	152	143	138	80	38,914
delivery rural (10)shape&pc	401,696	11,747	2,407	812	351	172	109	127	92	76	64	56	29	417,295
air/water trans. (14) weight	193,077	13,736	5,288	2,525	1,374	799	599	813	696	637	603	580	337	217,398
hwy/rail trans. (14)cube	149,099	11,326	4,932	2,487	1,390	838	630	857	724	666	628	603	349	170,702
Other weight	116,560	8,292	3,192	1,524	830	482	362	491	420	384	364	350	203	131,243
Total Cost	4,723,057	404,565	106,591	65,559	22,263	11,515	8,891	11,555	4,664	4,792	3,429	4,250	2,016	5,342,441
Total Unit Cost	$ 0.096	$ 0.275	$ 0.341	$ 0.609	$ 0.482	$ 0.518	$ 0.632	$ 0.703	$ 0.372	$ 0.472	$ 0.395	$ 0.560	$ 0.491	$ 0.104
number of additional ounces purchased		1	2	3	4	5	6	7	8	9	10	10	10	3,310,010,630
total number of additional ounces purchased		1,471,054,864	625,145,058	323,079,397	184,606,120	111,096,807	84,417,307	115,058,523	100,367,640	91,406,366	86,763,555	75,957,847	41,057,146	$ 455,162,636
cost of pieces in excess of first ounce cost		$ 263,536,723	$ 76,625,266	$ 55,294,275	$ 17,838,896	$ 9,384,581	$ 7,542,503	$ 9,979,143	$ 3,480,967	$ 3,818,713	$ 2,597,119	$ 3,522,075	$ 1,622,374	0.1375
Marginal Cost Difference		$ 0.179	$ 0.066	$ 0.268	$ (0.126)	$ 0.036	$ 0.114	$ 0.071	$ (0.331)	$ 0.100	$ (0.077)	$ 0.164	$ (0.069)	

Table 3:
Costs by Ounce Increment for Outside County Periodicals

	0 to 1	1 to 2	2 to 3	3 to 5	5 to 6	6 to 7	7 to 9	9 to 13	over 13	Total
volume	573,914,658	1,000,902,421	893,545,710	1,532,163,117	1,078,425,983	1,156,428,605	1,371,867,042	1,361,326,382	1,259,093,091	10,227,667,007
pounds	18,376,748	95,098,735	133,446,890	380,762,268	368,446,896	464,508,120	668,054,022	907,840,440	1,342,752,184	4,379,286,305
cubic feet (weight/density)	1,071,316	5,450,329	8,123,065	23,323,626	22,588,895	28,423,070	40,914,970	55,511,162	82,380,580	267,787,013
all mp (3.1) tally	45,050	99,310	102,422	255,594	94,458	77,517	122,861	127,636	182,333	1,107,182
window service (3.2) tally	589	414	149	151	0	333	0	0	432	2,069
delivery in-office (6.1) tally	12,694	29,822	30,771	83,273	28,539	21,131	31,297	28,254	30,885	296,666
delivery in-office (6.2) 6.1	2,317	5,443	5,616	15,199	5,209	3,857	5,712	5,157	5,637	54,147
del. route (7.1) piece	1,295	2,258	2,016	3,456	2,433	2,609	3,095	3,071	2,840	23,071
del. access (7.2) piece	904	1,577	1,408	2,414	1,699	1,822	2,162	2,145	1,984	16,117
elem. load (7.3)shape&pc	5,791	10,495	6,988	12,232	8,704	8,878	10,803	10,189	9,977	84,057
del. support (7.4) sum6&7	3,624	7,737	7,217	17,764	7,250	6,035	8,298	7,650	8,006	73,582
vehicle service (8) cube	286	1,453	2,166	6,218	6,022	7,578	10,908	14,800	21,963	71,394
delivery rural (10)shape&pc	10,212	16,334	17,072	29,852	21,086	22,378	26,686	26,209	24,524	194,353
air/water trans. (14) weight	104	540	758	2,162	2,092	2,638	3,794	5,155	7,625	24,868
hwy/rail trans. (14)cube	1,285	6,536	9,741	27,969	27,088	34,084	49,064	66,567	98,789	321,123
Other weight	91	470	660	1,882	1,831	2,296	3,302	4,487	6,837	21,845
Total Cost	84,241	182,369	186,983	456,187	206,402	191,156	277,982	301,320	401,632	2,290,273
Total Unit Cost	$ 0.147	$ 0.182	$ 0.209	$ 0.298	$ 0.191	$ 0.165	$ 0.203	$ 0.221	$ 0.319	$ 0.224
Presort Adj. (USPS LR-I -94)	(0.038)	(0.006)	(0.007)	(0.008)	0.011	0.016	0.002	$ (0.001)	(0.003)	$ -
Adjusted Unit Cost	$ 0.109	$ 0.176	$ 0.202	$ 0.290	$ 0.203	$ 0.181	$ 0.204	$ 0.220	$ 0.316	$ 0.224
Marginal Cost Difference		$ 0.065	$ 0.027	$ 0.088	$ (0.088)	$ 0.018	$ 0.023	$ 0.016	$ 0.096	

Table 3:
Costs by Ounce Increment for Outside County Periodicals

	0 to 1	1 to 2	2 to 3	3 to 5	5 to 6	6 to 7	7 to 9	9 to 13	over 13	Total	
volume	573,914,658	1,000,902,421	893,545,710	1,532,163,117	1,078,425,983	1,156,428,605	1,371,867,042	1,361,326,382	1,259,093,091	10,227,667,007	
pounds	18,376,748	95,098,735	133,446,890	380,762,268	368,446,896	464,508,120	668,054,022	907,840,440	1,342,752,184	4,379,286,305	
cubic feet (weight/density)	1,071,316	5,450,329	8,123,065	23,323,626	22,588,895	28,423,070	40,914,970	55,511,162	82,380,580	267,787,013	
all mp (3.1) tally	45,050	99,310	102,422	255,594	94,458	77,517	122,861	127,636	182,333	1,107,182	
window service (3.2) tally	589	414	149	151	0	333	0	0	432	2,069	
delivery in-office (6.1) tally	12,694	29,822	30,771	83,273	28,539	21,131	31,297	28,254	30,885	296,666	
delivery in-office (6.2) 6.1	2,317	5,443	5,616	15,199	5,209	3,857	5,712	5,157	5,637	54,147	
del. route (7.1) piece	1,295	2,258	2,016	3,456	2,433	2,609	3,095	3,071	2,840	23,071	
del. access (7.2) piece	904	1,577	1,408	2,414	1,699	1,822	2,162	2,145	1,984	16,117	
elem. load (7.3)shape&pc	5,791	10,495	6,988	12,232	8,704	8,878	10,803	10,189	9,977	84,057	
del. support (7.4) sum6&7	3,624	7,737	7,217	17,764	7,250	6,035	8,298	7,650	8,006	73,582	
vehicle service (8) cube	286	1,453	2,166	6,218	6,022	7,578	10,908	14,800	21,963	71,394	
delivery rural (10)shape&pc	10,212	16,334	17,072	29,852	21,086	22,378	26,686	26,209	24,524	194,353	
air/water trans. (14) weight	104	540	758	2,162	2,092	2,638	3,794	5,155	7,625	24,868	
hwy/rail trans. (14)cube	1,285	6,536	9,741	27,969	27,088	34,084	49,064	66,567	98,789	321,123	
Other weight	91	470	660	1,882	1,821	2,296	3,302	4,487	6,637	21,645	
Total Cost	84,241	182,389	186,983	458,167	208,402	191,158	277,982	301,320	401,632	2,290,273	1,944,282
Total Unit Cost	$ 0.147	$ 0.182	$ 0.209	$ 0.299	$ 0.191	$ 0.165	$ 0.203	$ 0.221	$ 0.319	$ 0.224	$ 0.190
Presort Adj. (USPS LR-I -94)	(0.038)	(0.006)	(0.007)	(0.003)	0.011	0.016	0.002	(0.001)	(0.003)	$ -	
Adjusted Unit Cost	$ 0.109	$ 0.176	$ 0.202	$ 0.296	$ 0.203	$ 0.181	$ 0.204	$ 0.220	$ 0.316	$ 0.224	
Marginal Cost Difference		$ 0.035	$ 0.027	$ 0.090	$ (0.106)	$ 0.011	$ 0.056	$ 0.019	$ 0.096		

Table 3:
Costs by Ounce Increment for Periodicals

	0 to 1	1 to 2	2 to 3	3 to 5	5 to 6	6 to 7	7 to 9	9 to 13	over 13	Total
volume	573,914,658	1,000,902,421	893,545,710	1,532,163,117	1,078,425,983	1,156,428,605	1,371,867,042	1,361,326,382	1,259,093,091	10,227,667,007
pounds	18,376,748	95,098,735	133,446,890	380,762,268	368,446,896	464,508,120	668,054,022	907,840,440	1,342,752,184	4,379,286,305
cubic feet (weight/density)	1,071,316	5,450,329	8,123,065	23,323,626	22,588,895	28,423,070	40,914,970	55,511,162	82,380,580	267,787,013
all mp (3.1) tally	45,050	99,310	102,422	255,594	94,458	77,517	122,861	127,636	182,333	1,107,182
window service (3.2) tally	589	414	149	151	0	333	0	0	432	2,069
delivery in-office (6.1) tally	12,694	29,822	30,771	83,273	28,539	21,131	31,297	28,254	30,885	296,666
delivery in-office (6.2) 6.1	2,317	5,443	5,616	15,199	5,209	3,857	5,712	5,157	5,637	54,147
del. route (7.1) piece	1,295	2,258	2,016	3,456	2,433	2,609	3,095	3,071	2,840	23,071
del. access (7.2) piece	*904*	*1,577*	*1,408*	*2,414*	*1,699*	*1,822*	*2,162*	*2,145*	*1,984*	*16,117*
elem. load (7.3)shape&wt	1,046	6,066	2,785	6,978	6,758	8,037	11,884	15,370	25,132	84,057
del. support (7.4) sum6&7	2,793	6,961	6,480	16,844	6,909	5,888	8,488	8,558	10,661	73,582
vehicle service (8) cube	286	1,453	2,166	6,218	6,022	7,578	10,908	14,800	21,963	71,394
delivery rural (10)shape&pc	10,212	16,334	17,072	29,852	21,086	22,378	26,686	26,209	24,524	194,353
air/water trans. (14) weight	104	540	758	2,162	2,092	2,638	3,794	5,155	7,625	24,868
hwy/rail trans. (14)cube	1,285	6,536	9,741	27,969	27,088	34,084	49,064	66,567	98,789	321,123
Other weight	94	488	685	1,953	1,890	2,383	3,427	4,657	6,888	22,464
Total Cost	78,669	177,202	182,067	452,064	204,184	190,255	279,378	307,579	419,694	2,291,092
Total Unit Cost	$ 0.137	$ 0.177	$ 0.204	$ 0.295	$ 0.189	$ 0.165	$ 0.204	$ 0.226	$ 0.333	$ 0.224
Presort Adj. (USPS LR-I -94)	(0.038)	(0.006)	(0.007)	(0.003)	0.011	0.016	0.002	(0.001)	(0.003)	$ -
Adjusted Unit Cost	$ 0.099	$ 0.171	$ 0.197	$ 0.292	$ 0.201	$ 0.180	$ 0.205	$ 0.225	$ 0.330	$ 0.224
Marginal Cost Difference		$ 0.040	$ 0.027	$ 0.091	$ (0.106)	$ 0.014	$ 0.061	$ 0.022	$ 0.107	

Table 4 (revised):
Estimated Test Year Unit Costs for Piece-Rated and Pound-Rated Standard Mail

		REG	ECR
All Shapes	< 3.0 oz	$ 0.1249	$ 0.0675
	> 3.0 oz	$ 0.2562	$ 0.0826
	< 3.5 oz	$ 0.1274	$ 0.0683
	> 3.5 oz	$ 0.2808	$ 0.0838
	average	$ 0.1482	$ 0.0721
Letters	< 3.0 oz	$ 0.0939	$ 0.0655
	> 3.0 oz	$ 0.3138	$ 0.1549
	< 3.5 oz	$ 0.0944	$ 0.0659
	> 3.5 oz	$ 1.4773	$ 0.2420
	average	$ 0.0962	$ 0.0668
Flats	< 3.0 oz	$ 0.2723	$ 0.0675
	> 3.0 oz	$ 0.2015	$ 0.0784
	< 3.5 oz	$ 0.2508	$ 0.0684
	> 3.5 oz	$ 0.2105	$ 0.0794
	average	$ 0.2337	$ 0.0724
Parcels	< 3.0 oz	$ 1.9267	$ 2.1703
	> 3.0 oz	$ 0.9424	$ 5.5090
	< 3.5 oz	$ 1.9651	$ 2.4066
	> 3.5 oz	$ 0.9242	$ 5.2508
	average	$ 1.0249	$ 3.3125
Flat + Parcel	< 3.0 oz	$ 0.2859	$ 0.0693
	> 3.0 oz	$ 0.2538	$ 0.0814
	< 3.5 oz	$ 0.2687	$ 0.0702
	> 3.5 oz	$ 0.2730	$ 0.0826
	average	$ 0.2679	$ 0.0747

Table 4 (revised):
Estimated Test Year Unit Costs for Piece-Rated and Pound-Rated Standard Mail

		REG	[redacted]
All Shapes	< 3.0 oz	$ 0.1250	[redacted]
	> 3.0 oz	$ 0.2565	[redacted]
	< 3.5 oz	$ 0.1274	[redacted]
	> 3.5 oz	$ 0.2810	[redacted]
	average	$ 0.1483	[redacted]
Letters	< 3.0 oz	$ 0.0939	[redacted]
	> 3.0 oz	$ 0.3139	[redacted]
	< 3.5 oz	$ 0.0945	[redacted]
	> 3.5 oz	$ 1.4775	[redacted]
	average	$ 0.0962	[redacted]
Flats	< 3.0 oz	$ 0.2724	[redacted]
	> 3.0 oz	$ 0.2018	[redacted]
	< 3.5 oz	$ 0.2509	[redacted]
	> 3.5 oz	$ 0.2108	[redacted]
	average	$ 0.2339	[redacted]
Parcels	< 3.0 oz	$ 1.9268	[redacted]
	> 3.0 oz	$ 0.9429	[redacted]
	< 3.5 oz	$ 1.9652	[redacted]
	> 3.5 oz	$ 0.9246	[redacted]
	average	$ 1.0253	[redacted]
Flat + Parcel	< 3.0 oz	$ 0.2860	[redacted]
	> 3.0 oz	$ 0.2540	[redacted]
	< 3.5 oz	$ 0.2638	[redacted]
	> 3.5 oz	$ 0.2733	[redacted]
	average	$ 0.2680	[redacted]

Table 4 (revised):
Estimated Test Year Unit Costs for Piece-Rated and Pound-Rated Standard Mail

		REG	ECR
All Shapes	< 3.0 oz	$ 0.1249	$ 0.0675
	> 3.0 oz	$ 0.2562	$ 0.0826
	< 3.5 oz	$ 0.1274	$ 0.0683
	> 3.5 oz	$ 0.2808	$ 0.0838
	average	$ 0.1482	$ 0.0721
Letters	< 3.0 oz	$ 0.0939	$ 0.0655
	> 3.0 oz	$ 0.3138	$ 0.1549
	< 3.5 oz	$ 0.0944	$ 0.0659
	> 3.5 oz	$ 1.4773	$ 0.2420
	average	$ 0.0962	$ 0.0668
Flats	< 3.0 oz	$ 0.2723	$ 0.0675
	> 3.0 oz	$ 0.2015	$ 0.0784
	< 3.5 oz	$ 0.2508	$ 0.0684
	> 3.5 oz	$ 0.2105	$ 0.0794
	average	$ 0.2337	$ 0.0724
Parcels	< 3.0 oz	$ 1.9267	$ 2.1703
	> 3.0 oz	$ 0.9424	$ 5.5090
	< 3.5 oz	$ 1.9651	$ 2.4066
	> 3.5 oz	$ 0.9242	$ 5.2508
	average	$ 1.0249	$ 3.3125
Flat + Parcel	< 3.0 oz	$ 0.2859	$ 0.0693
	> 3.0 oz	$ 0.2538	$ 0.0814
	< 3.5 oz	$ 0.2637	$ 0.0702
	> 3.5 oz	$ 0.2730	$ 0.0826
	average	$ 0.2679	$ 0.0747

Table 4 (revised):
Estimated Test Year Unit Costs for Piece-Rated and Pound-Rated Standard Mail

		REG	EGR
All Shapes	< 3.0 oz	$ 0.1250	$ 0.0675
	> 3.0 oz	$ 0.2565	$ 0.0826
	< 3.5 oz	$ 0.1274	$ 0.0683
	> 3.5 oz	$ 0.2810	$ 0.0838
	average	$ 0.1483	$ 0.0721
Letters	< 3.0 oz	$ 0.0939	$ 0.0655
	> 3.0 oz	$ 0.3139	$ 0.1549
	< 3.5 oz	$ 0.0945	$ 0.0659
	> 3.5 oz	$ 1.4775	$ 0.2420
	average	$ 0.0962	$ 0.0668
Flats	< 3.0 oz	$ 0.2724	$ 0.0675
	> 3.0 oz	$ 0.2018	$ 0.0784
	< 3.5 oz	$ 0.2509	$ 0.0684
	> 3.5 oz	$ 0.2108	$ 0.0794
	average	$ 0.2339	$ 0.0724
Parcels	< 3.0 oz	$ 1.9268	$ 2.1703
	> 3.0 oz	$ 0.9429	$ 5.5090
	< 3.5 oz	$ 1.9652	$ 2.4066
	> 3.5 oz	$ 0.9246	$ 5.2508
	average	$ 1.0253	$ 3.3125
Flat + Parcel	< 3.0 oz	$ 0.2860	$ 0.0693
	> 3.0 oz	$ 0.2540	$ 0.0814
	< 3.5 oz	$ 0.2638	$ 0.0702
	> 3.5 oz	$ 0.2733	$ 0.0826
	average	$ 0.2680	$ 0.0747

Table 4:
Estimated Test Year Unit Costs for Piece-Rated and Pound-Rated Standard Mail

		REG	ECR
All Shapes	< 3.0 oz	$ 0.1250	$ 0.0750
	> 3.0 oz	$ 0.2565	$ 0.0861
	< 3.5 oz	$ 0.1274	$ 0.0756
	> 3.5 oz	$ 0.2810	$ 0.0872
	average	$ 0.1483	$ 0.0786
Letters	< 3.0 oz	$ 0.0939	$ 0.0750
	> 3.0 oz	$ 0.3139	$ 0.1762
	< 3.5 oz	$ 0.0945	$ 0.0755
	> 3.5 oz	$ 1.4775	$ 0.2730
	average	$ 0.0962	$ 0.0766
Flats	< 3.0 oz	$ 0.2724	$ 0.0729
	> 3.0 oz	$ 0.2018	$ 0.0818
	< 3.5 oz	$ 0.2509	$ 0.0736
	> 3.5 oz	$ 0.2108	$ 0.0827
	average	$ 0.2339	$ 0.0770
Parcels	< 3.0 oz	$ 1.9268	$ 2.2694
	> 3.0 oz	$ 0.9429	$ 8.3123
	< 3.5 oz	$ 1.9652	$ 2.5205
	> 3.5 oz	$ 0.9246	$ 8.0432
	average	$ 1.0263	$ 3.7867
Flat + Parcel	< 3.0 oz	$ 0.2860	$ 0.0749
	> 3.0 oz	$ 0.2540	$ 0.0847
	< 3.5 oz	$ 0.2638	$ 0.0756
	> 3.5 oz	$ 0.2733	$ 0.0859
	average	$ 0.2680	$ 0.0795

Table 5 (revised): Test Year 2003 Standard Mail Costs by Shape

	Cost per Piece (cents)
Parcels	107.53
Flats	13.93
Difference	93.60

Table 5 (revised): Test Year 2003 Standard Mail Costs by Shape

	Cost per Piece (cents)
Parcels	107.57
Flats	13.93
Difference	93.63

Table 5 (revised): Test Year 2003 Standard Mail Costs by Shape

	Cost per Piece (cents)
Parcels	107.53
Flats	13.93
Difference	93.60

Table 5 (revised): Test Year 2003 Standard Mail Costs by Shape

	Cost per Piece (cents)
Parcels	107.57
Flats	13.93
Difference	93.63

Table 5: Test Year 2003 Standard Mail Costs by Shape

	Cost per Piece (cents)
Parcels	107.86
Flats	14.43
Difference	93.43

Table 6 (Previously Attachment F - Table 4 in USPS-T-27/R2000-1)
Calculation of Cost Difference Due to Differences in Presorting and Drop Shipment
FY 2000 Standard Mail

1) Weight by Entry Discount (Attachment F, Table 5 in USPS-T-27/R2000-1)

	None	BMC	SCF	DDU	Total
Flats	1,602,204	1,603,859	3,640,710	1,461,540	8,308,313
Parcels	283,293	96,322	36,379	384	416,379

2) Cost Avoidance $/lb (USPS-LR-J-68)

None	BMC	SCF	DDU
0	0.117	0.147	0.185

3) Avoided Costs (= (1) * (2))

	None	BMC	SCF	DDU	Total	Average Avoided Cost/Piece
Flats	0	188,348	536,062	270,183	994,594	0.025
Parcels	0	11,312	5,357	71	16,739	0.023

4) Pieces by Presort Level (Attachment F, Tables 1 and 2 in USPS-T-27/R2000-1))

	Basic	3/5 Digit	Carrier	125 Walk	Saturation	Total
Flats	1,295,599	14,476,245	12,879,082	1,535,675	9,376,070	39,562,671
Parcels	211,642	500,111	14,383	254	2,488	728,878

5) Presort Cost Avoidances $ / pc

Basic	3/5 Digit	Carrier	125 Walk	Saturation
0	0.07276435	0.18217647	0.21495684	0.22363219

6) Avoided Costs (= (4) * (5))

	Basic	3/5 Digit	Carrier	125 Walk	Saturation	Total	Average Avoided Cost/ Piece
Flats	0	1,053,355	2,346,266	330,104	2,096,791	5,826,515	0.147
Parcels	0	36,390	2,620	55	556	39,622	0.054

7) Cost Difference Due to Differences in Entry and Presort Profile

Flats

7a)	0.002	$ / piece saved due to entry profile relative to parcels. (= (3a) - (3b))
7b)	0.093	$ / piece saved due to presort profile relative to parcels. (= (6a) - (6b))
7c)	0.095	$ / piece of difference in average costs of flats and parcels are explained by differences in presorting and entry profiles. (= (7a) + (7b))

Table 7: Summary of Standard ECR (Commercial) by Destination Entry

Basic Tier	ounce increment												
	4.0-5.0	5.0-6.0	6.0-7.0	7.0-8.0	8.0-9.0	9.0-10.0	10.0-11.0	11.0-12.0	12.0-13.0	13.0-14.0	14.0-15.0	15.0-16.0	Total 4.0 - 16.0
None	62,947,178	42,860,604	40,372,902	15,626,310	5,926,242	4,416,590	2,675,955	2,534,023	1,832,079	1,357,595	1,429,404	901,882	182,880,764
DBMC	304,562,198	120,119,789	63,075,825	27,138,430	22,021,579	10,774,357	6,329,973	7,702,567	2,799,798	3,022,902	2,017,309	1,584,313	571,149,040
DSCF	1,393,019,957	687,128,369	347,200,787	176,319,894	148,814,454	72,353,137	36,414,942	50,964,651	21,368,004	11,172,024	3,382,548	6,673,169	2,954,811,935
DDU	37,398,881	25,043,536	15,959,875	10,903,164	7,860,762	4,209,715	2,063,488	1,574,426	845,287	493,026	303,305	382,187	107,037,653

High Density Tier	ounce increment												
	4.0-5.0	5.0-6.0	6.0-7.0	7.0-8.0	8.0-9.0	9.0-10.0	10.0-11.0	11.0-12.0	12.0-13.0	13.0-14.0	14.0-15.0	15.0-16.0	Total 4.0 - 16.0
None	1,636,835	890,113	569,956	298,979	178,926	109,479	40,879	38,142	32,787	14,232	12,689	12,137	3,835,154
DBMC	126,873	121,490	16,515	59,319	4,802	14,802	972	0	5,895	0	23,327	0	373,995
DSCF	60,795,091	72,710,377	40,217,767	21,260,749	8,542,132	4,873,267	1,894,657	1,173,116	696,705	384,835	249,103	133,647	212,931,445
DDU	102,934,800	84,469,856	65,354,650	42,426,628	26,980,279	17,831,244	9,889,784	6,391,892	3,224,312	2,101,482	1,087,436	506,828	363,199,190

Saturation Tier	ounce increment												
	4.0-5.0	5.0-6.0	6.0-7.0	7.0-8.0	8.0-9.0	9.0-10.0	10.0-11.0	11.0-12.0	12.0-13.0	13.0-14.0	14.0-15.0	15.0-16.0	Total 4.0 - 16.0
None	15,905,373	8,600,673	4,984,591	3,250,802	1,904,023	1,200,882	741,149	564,954	392,133	271,079	165,269	201,224	38,182,153
DBMC	2,094,842	338,296	140,930	115,936	58,777	78,336	70,563	169,359	79,829	62,743	71,212	74,034	3,354,858
DSCF	188,741,001	123,337,983	47,584,828	17,671,856	7,997,248	3,740,679	2,120,129	819,056	520,678	309,605	317,701	136,121	393,296,888
DDU	876,540,999	540,959,107	198,826,692	97,598,248	33,296,442	15,316,674	5,964,206	3,263,994	1,923,344	1,030,001	491,103	396,843	1,775,607,654
Total (all tiers)	3,046,704,029	1,706,580,192	824,305,318	412,670,315	263,585,666	134,919,161	68,206,696	75,196,180	33,720,852	20,219,526	9,550,406	11,002,385	6,606,660,728

Appendix A: Program Documentation

A. Computer Hardware and Software

The IOCS data processing is performed on a Data General AViiON minicomputer with four Pentium Pro microprocessors and one gigabyte of RAM, running the DGUX version of UNIX operating system. Source programs ending with an ".f" file extension are FORTRAN programs and programs ending in a ".sm" file extension are SORT/MERGE programs. The remaining data processing is performed in Excel workbooks (.xls file extension) on PCs running the Microsoft Windows NT 4 and Windows 2000 operating systems and Microsoft Office.

B. Preparation of the IOCS Data

The following programs are used to extract, code, and process the 2000 IOCS data set in preparation for the proposed Postal Service method volume-variable cost distribution for both mail processing and administration/window service costs for clerks and mailhandlers.

Program: **cadoc00by_rep.f** - Divides IOCS clerk/mailhandler tallies by office group (MODS 1&2, BMCs, Non-MODS) and assigns the tallies to cost pools

Input: **FY00 IOCS Data** (USPS-LR-J-10)
mods_usps.00 - List of MODS 1&2 finance numbers used to identify MODS 1&2 offices

Output: **mods12_mp00by_new.dat** – IOCS mail processing tallies for MODS 1&2 offices
mods12_aw00by_new.dat – IOCS administrative and window service tallies for MODS 1&2 offices
bmcs_mp00by_new.dat – IOCS mail processing tallies for BMCs
bmcs_aw00by_new.dat – IOCS administrative and window service tallies for BMCs
nonmods_mp00by_new.dat – IOCS mail processing tallies for Non-MODS offices
nonmods_aw00by_new.dat – IOCS administrative and window service tallies for Non-MODS offices
nonmods_op88_new.dat – IOCS expedited delivery tallies for Non-MODS offices

C. Postal Service Method Volume-Variable Cost Estimates by Weight Increment– Clerks and Mailhandlers, Mail Processing

The volume-variable cost distribution FORTRAN programs replicate the function of the mail processing cost distribution SAS programs documented in USPS-LR-J-55. The FORTRAN programs described below divide the cost estimates by subclass, cost pool, shape of mail, and weight category. Weight categories are by the half-ounce increment up to four ounces, by whole ounce increment up to 16 ounces, and a final category of over 16 ounces. Tallies are assigned to a weight category using the IOCS question 23G, whose response is reported in IOCS fields F165, F166, and F167. The results of these programs are exported into Microsoft Excel where final results are summarized and reported.

Program: **modsproc00_wgt.f** – Estimates mail processing volume-variable costs for MODS 1&2 offices by activity code, cost pool, and weight increment

Input: **mods12_mp00by_new.dat** – IOCS mail processing tallies for MODS 1&2 offices
iocs2000.h – Declaration of IOCS tally fields
activity00.ecr.cra – List of the direct and class specific mixed activity codes
mixclass.intl – List of class specific mixed mail activity codes
mxmail.intl.dat – Maps the direct activity codes to their respective class specific mixed mail activity codes
costpools.00.619 - List of MODS 1&2 cost pool dollars and corresponding variability factors used in the cost distribution for MODS 1&2 offices (USPS-LR-J-55)
windk_wgt_ecr.00.619 – Distributed clerk and mailhandler window service costs by activity code and weight increment for 'function 4 support' cost pool distribution (USPS-LR-J-55) - see Section C below

Output: **mods002by.data** – Estimated mail processing volume-variable costs by cost pool, activity code, and weight increment for MODS 1&2 offices

Program: **sumclass_mod_wgt.f** - Rolls up the output from modsproc00_wgt.f from activity code to subclass by cost pool, shape, and weight increment

Input: **mods002by.data** – Estimated mail processing volume-variable costs by cost pool, activity code, and weight increment for MODS 1&2 offices
costpools.00.619 - List of cost pools for MODS 1&2 offices (USPS-LR-J-55)
activity00.ecr.cra – List of the direct and class specific mixed activity codes
classes_intl.cra.new - List of new CRA subclasses
classmap_intl.new - Maps IOCS activity codes to the appropriate CRA subclass

Output: **mod00_wgt2.csv** – Estimated mail processing volume-variable costs for selected subclasses by cost pool, shape, and weight category for MODS 1&2 offices

Program: **bmcproc00_wgt.f** – Estimates mail processing volume-variable costs for BMCs by activity code, cost pool, and weight increment

Input: **bmcs_mp00by_new.dat** – IOCS mail processing tallies for BMCs
iocs2000.h – Declaration of IOCS tally fields
activity00.ecr.cra – List of the direct and class specific mixed activity codes
mixclass.intl – List of class specific mixed mail activity codes
mxmail.intl.dat – Maps the direct activity codes to their respective class specific mixed mail activity codes
costpools.00.bmc.619 – List of BMC cost pool dollars and corresponding variability factors used in the cost distribution for BMCs (USPS-LR-J-55)

Output: **bmc002by_wgt.data** – Estimated mail processing volume-variable costs by cost pool, activity code, and weight increment for BMCs

Program: **sumclass_bmc_wgt.f** - Rolls up the output from bmcproc00_wgt.f from activity code to subclass by cost pool, shape, and weight increment

Input: **bmc002by_wgt.data** – Estimated mail processing volume-variable costs by cost pool, activity code, and weight increment for BMCs
costpools.00.bmc.619 – List of cost pools for BMCs (USPS-LR-J-55)
activity00.ecr.cra – List of the direct and class specific mixed activity codes
classes_intl.cra.new - List of new CRA subclasses
classmap_intl.new - Maps IOCS activity codes to the appropriate CRA subclass

Output: **bmc00_wgt2.csv** – Estimated mail processing volume-variable costs for selected subclasses by cost pool, shape, and weight category for BMCs

Program: **nmodproc00_wgt.f** – Estimates mail processing volume-variable costs for Non-MODS offices by activity code, cost pool, and weight increment

Input: **nonmods_mp00by_new.dat** – IOCS mail processing tallies for Non-MODS offices
iocs2000.h – Declaration of IOCS tally fields
activity00.ecr.cra – List of the direct and class specific mixed activity codes
mixclass.intl – List of class specific mixed mail activity codes
mxmail.intl.dat – Maps the direct activity codes to their respective class specific mixed mail activity codes
costpools.00.nmod.619 – List of Non-MODS office cost pool dollars and corresponding variability factors used in the cost distribution for Non-MODS offices (USPS-LR-J-55)

Output: **nmod00by_wgt.data** – Estimated mail processing volume-variable costs by cost pool, activity code, and weight increment

Program: **sumclass_nmod_wgt.f** - Rolls up the output from nmodproc00_wgt.f from activity code to subclass by cost pool, shape, and weight increment

Input: **nmod00by_wgt.data** – Estimated mail processing volume-variable costs by cost pool, activity code, and weight increment
costpools.00.nmod.619 – List of cost pools for Non-MODS offices (USPS-LR-J-55)
activity00.ecr.cra – List of the direct and class specific mixed activity codes
classes_intl.cra.new - List of new CRA subclasses
classmap_intl.new - Maps IOCS activity codes to the appropriate CRA subclasses

Output: **nmod00_wgt2.csv** – Estimated mail processing volume-variable costs for selected subclasses by cost pool, shape, and weight category

Workbook: **Mail Proc by Wgt BY00 New.xls** – Summarizes the BY00 mail processing volume-variable cost estimates for all offices by subclass, cost pool, shape, and weight increment. Applies test year piggyback factors, cost ratios, and reconciliation factors to convert base year mail processing costs to test year mail processing cost to be used in the weight increment analysis

Input: **mod00_wgt2.csv** – Estimated mail processing volume-variable costs for selected subclasses by cost pool, shape, and weight category for MODS 1&2 offices
bmc00_wgt2.csv – Estimated mail processing volume-variable costs for selected subclasses by cost pool, shape, and weight category for BMCs
nmod00_wgt2.csv – Estimated mail processing volume-variable costs for selected subclasses by cost pool, shape, and weight category for Non-MODS offices
Test Year 03 Piggyback Factors, Cost Ratios, and Reconciliation Factors –USPS-LR-J-52

D. Postal Service Method Cost Estimates by Weight Increment– Clerks and Mailhandlers, Window Service

The window service cost distribution FORTRAN programs replicate the function of the ADMWIN SAS programs (USPS-LR-J-55). The FORTRAN programs described below divide the cost estimates by subclass, cost pool, shape of mail, and weight category. Weight categories are by the half-ounce increment up to four ounces, by whole ounce increment up to 16 ounces, and a final category of over 16 ounces. Costs are assigned to a weight category using the IOCS question 23G, whose response are located in IOCS fields F165, F166, and F167. The results of these programs are exported into Microsoft Excel where final results are summarized and reported.

Program: **admwin_set.f** – *Prepares administration and window service IOCS tallies for cost* distribution. Converts tally dollar values (F9250) to cost pool dollars, assigns the tally to a CAG category, and encircles activity codes

Input: **fincag.98** – List of tally finance numbers and CAG
iocs2000.h – Declaration of IOCS tally fields
mods12_aw00by_new.dat – IOCS administrative and window service tallies for MODS 1&2 offices
bmcs_aw00by_new.dat – IOCS administrative and window service tallies for BMCs
nonmods_aw00by_new.dat – IOCS administrative and window service tallies for Non-MODS offices

Output: **admwin00.dat** - Administrative and window service tallies used for cost distribution for all office types

Program: **admwin_wgt2.f** – Estimates the distributed volume-variable costs for administration and window service clerks and mailhandlers by weight increment

Input: **activity00.auto.intl2** – List of activity codes and corresponding subclass category codes
iocs2000.h – Declaration of IOCS tally fields
admwin00.dat - Administrative and window service tallies used for cost distribution for all office types

Output: **awdist00_wgt.data** – Estimated administrative/window service volume-variable costs by cost segment, activity code, and weight increment

Program: **sumclass_wgt.f** - Rolls up the window service volume-variable costs estimated by admwin_wgt2.f from activity code to subclass by weight increment

Input: **wgtinc.prn2** – List of weight increment categories
activity00.auto.intl2 – List of activity codes and corresponding subclass category codes
classes.auto.intl2 - *List of CRA subclasses*
awdist00_wgt.data – Estimated administrative/window service volume-variable costs by cost segment, activity code, and weight increment

Output: **wincost_wgt00.csv** – Estimated window service direct labor volume-variable cost estimates by subclass, shape, and weight increment.

Workbook: **Volumes by Wgt GFY00 update.xls** – GFY00 RPW volumes and weights by weight increment, shape, and various subclasses

Input: **GFY00 RPW Volumes and Weights** – USPS-LR-J-112

Workbook: **Win Key Fcn4 00.xls** – Distributes BY00 CRA window service costs (CRA Cost Segment 3.2.1) to shape

Input: **wincost_wgt00.csv** – Estimated window service direct labor volume-variable cost estimates by subclass, shape, and weight increment
BY00 CRA Window Service Costs – CRA worksheet 3.2.1 (USPS-LR-J-57)
GFY00 RPW Volumes – RPW volumes by shape from the file 'Volumes by Wgt GFY00 update.xls'

Workbook: **win cost by oz 00 new.xls** – Uses the estimated window service volume-variable costs as a distribution key to distribute BY00 CRA window service direct labor costs to weight increment

Input: **wincost_wgt00.csv** – Distributed window service direct labor volume-variable cost estimates by subclass, shape, and weight increment
BY00 CRA Window Service Direct Labor Costs – Costs by subclass and shape created in the file 'Win Key Fcn4 00.xls'

Output: **windkecr00.prn** – BY00 CRA C/S 3.1.2 costs by activity code and weight increment – worksheet 'ECR Actv'

Workbook: **sm cost by oz 00 new.xls** – Uses RPW controlled volumes as a distribution key to distribute BY00 CRA window service stamp sales/meter setting volume-variable costs to weight increment

Input: **BY00 CRA Window Service Stamped/Metered costs** – Costs by subclass and shape created in the file 'Win Key Fcn4 00.xls'
BY00 RPW Volumes – Volumes by subclass, shape, and weight increment ('Volumes by Wgt GFY00 update.xls')

Workbook: **Win Wgt BY00 New.xls** – Summarizes the direct labor and stamp sales/meter setting window service volume-variable costs into total window service costs by subclass, shape, and weight increment

Input: **win cost by oz 00 new.xls** – Window service direct labor volume-variable costs by subclass, shape, and weight increment
sm cost by oz 00 new.xls – Window service stamp sales/meter setting volume-variable costs by subclass, shape, and weight increment

Program: **win_key_ecr.f** – Creates the file of window service costs (CRA W/S 3.2.1) by activity code and weight increment used as a distribution key for the Function 4 Support cost pools. Used for both this weight increment analysis and the Mail Processing Cost Savings for Standard ECR study (USPS-LR-J-59)

Input: **activity00.ecr.cra** – List of the direct and class specific mixed activity codes
windkecr00.prn – BY00 CRA C/S 3.1.2 costs from the Microsoft Excel workbook 'win cost by oz 00 new.xls'

Output: **windk_wgt_ecr.00.619** – BY00 CRA Cost Segment 3.2 costs by activity code and weight increment for 'function 4 support' cost pool distribution (USPS-LR-J-55) – Used in Section B above

E. City Carrier In-Office Costs by Weight Increment

The following are descriptions of the FORTRAN programs used to replicate the LIOCATT cost distribution process for estimating CRA Cost Segment 6.1 City Carrier In-Office costs.

1. Preparation of the IOCS Data

The following programs are used to extract, code and process the 2000 IOCS data in preparation for LIOCATT distribution of city carrier in-office costs

Program: **encode_wgt.f** - Extracts the necessary data from the IOCS tally data set, encodes specific tally fields into indexes, and writes the indexes to be used by LIOCATT

Input: **FY00 IOCS tally data** (USPS-LR-J-10)
iocs2000.h – Declaration of IOCS tally fields
fincag.98 – Tally finance number and CAG combinations
activity00.ecr.all – List of activity codes

Output: **encdata** – Encoded IOCS tallies

Program: **encdata.sm** - Sorts the encoded IOCS data for the LIOCATT process

Input: **encdata**

Output: **encdata.s**

2. LIOCATT Based Cost Distribution Process

The LIOCATT distribution process is run through a main FORTRAN program, which runs FORTRAN subroutines that distribute mixed-mail costs. The declaration file 'liocatt.h' is included in each program and subroutine. This file contains common variables used by all programs and subroutines.

Program: **liocatt.f** - This program controls the LIOCATT process by running various FORTRAN programs, which replicates the LIOCATT process for mixed-mail cost distribution

Subroutines: **fillmixmap.f** - Produces a map for distributing the mixed mail codes to appropriate direct activity codes

Input: **activity00.ecr.all** - List of activity codes
mmcodes.intl – List of mixed-mail activity codes
mxmail.all.ecr - Maps class specific mixed-mail activity codes to *corresponding direct activity codes*

loaddata.f - Loads the encoded IOCS data

Input: **encdata.s** – Encoded IOCS data

fungroup.f - Forms function groups for operations

Input: **operrtemap** – Maps operation to function group

sortcost.f - Sort records for level 1 cost distribution

level1.f - Level 1 distribution of mixed-mail/not-handling tallies

level2.f - Level 2 distribution of mixed-mail/not-handling tallies

sortlev2a.f - Sort records for level 3 cost distribution

level3.f - Level 3 distribution of mixed-mail/not-handling tallies

report.f - Write results to file

Output: **level1b** – Level 1 distributed direct costs
level2b - Level 2 distributed direct costs
level3a – Level 3 direct costs
level3b – Level 3 distributed direct costs

3. Weight Increments

Program: **rpt_wgt22cra.f** – Summarized LIOCATT cost distribution results by subclass, shape, and weight increment for city carrier in-office costs

Input: **activity00.ecr.all** - List of activity codes and corresponding subclass codes
classes_ecr.old – List of old CRA subclasses
level1b – Level 1 distributed direct costs
level2b - Level 2 distributed direct costs
level3a – Level 3 direct costs
level3b – Level 3 distributed direct costs

Output: **car_wgt22_00cra2.csv** – Estimated City Carrier In-Office costs by subclass, shape, and weight increment

Workbook: **CC Costs BY00 New.xls** – Reports City Carrier In-Office costs by subclass, shape, and weight increment. Adjusts the FORTRAN replication of LIOCATT to match the BY00 CRA Cost Segment 6.1 costs

Input: **car_wgt22_00cra2.txt** – Estimated City Carrier In-Office costs by subclass, shape, and weight increment
BY00 CRA Cost Segment 6.1 Costs – CRA City Carrier In-Office costs from CS06&7.xls, worksheet 'oldoutputs' (USPS-LR-J-57)

F. Final Cost Estimates by Weight Increment

Workbook: **LR58ASP.xls** – Develops test year First-Class single piece unit costs by shape by weight increment.

Input: **Mail Proc by Wgt BY00 New.xls** – BY00 mail processing costs by shape by weight increment
Win Wgt BY00 New.xls – BY00 window service volume-variable costs by shape by weight increment
CC Costs BY00 New.xls – *BY00 City Carrier Costs by shape by weight increment*
USPS-LR-J-112 – BY00 volume and weight by shape by weight increment
USPS-LR-J-52 – Base year CRA costs and test year to base year cost ratio

Workbook: **LR58PRE.xls** – Develops test year First-Class presort unit costs by shape by weight increment.

Input: **Mail Proc by Wgt BY00 New.xls** – BY00 mail processing costs by shape by weight increment
Win Wgt BY00 New.xls – BY00 window service volume-variable costs by shape by weight increment
CC Costs BY00 New.xls – BY00 City Carrier Costs by shape by weight increment
USPS-LR-J-112 – BY00 volume and weight by shape by weight increment
USPS-LR-J-52 – Base year CRA costs and test year to base year cost ratio

Workbook: **LR58PER.xls** – Develops test year Periodicals unit costs by shape by weight increment.

Input: **Mail Proc by Wgt BY00 New.xls** – BY00 mail processing costs by shape by weight increment
Win Wgt BY00 New.xls – BY00 window service volume-variable costs by shape by weight increment
CC Costs BY00 New.xls – BY00 City Carrier Costs by shape by weight increment
USPS-LR-J-112 – BY00 volume and weight by shape by weight increment
USPS-LR-J-52 – Base year CRA costs and test year to base year cost ratio

Workbook: **LR58AREG.xls** – Develops test year Standard Regular Rate unit costs by shape by weight increment.

Input: **Mail Proc by Wgt BY00 New.xls** – BY00 mail processing costs by shape by weight increment
Win Wgt BY00 New.xls – BY00 window service volume-variable costs by shape by weight increment
CC Costs BY00 New.xls – BY00 City Carrier Costs by shape by weight increment
USPS-LR-J-112 – BY00 volume and weight by shape by weight increment
USPS-LR-J-52 – Base year CRA costs and test year to base year cost ratio

Workbook: **LR58AECR.xls** – Develops test year Standard ECR unit costs by shape by weight increment.

Input: **Mail Proc by Wgt BY00 New.xls** – BY00 mail processing costs by shape by weight increment
Win Wgt BY00 New.xls – BY00 window service volume-variable costs by shape by weight increment
CC Costs BY00 New.xls – BY00 City Carrier Costs by shape by weight increment
USPS-LR-J-112 – BY00 volume and weight by shape by weight increment
USPS-LR-J-52 – Base year CRA costs and test year to base year cost ratio

Workbook: **LR58STDCBS.xls** – Reports test year Standard Regular Rate and ECR unit costs by shape by weight increment. Calculates Standard Parcel/Flat cost difference.

Input: **LR58AREG.xls** – Test year Standard Regular Rate unit costs by shape by weight increment

Input: **LR58AECR.xls** – Test year Standard ECR unit costs by shape by weight increment

Workbook: **LR58ADJ.xls** – Calculation of test year cost difference due to differences in presorting and drop shipment for Standard Bulk mail.

Input: **USPS-LR-J-68** – Cost Avoidance

G. Standard ECR Volume Distribution by Destination Entry and Weight Increment

Workbook: **Volumes_tiers.xls** – Reports Standard ECR volumes from USPS-LR-J-112. Rolls up the volumes to develop volume distributions by destination entry and weight increment by rate element.

Input: **Standard ECR volumes** – from USPS-LR-J-112

Workbook: **Tiers_table.xls** – Reports volume distributions by destination entry and weight increment by rate element.

Input: **Volumes_tiers.xls** – volume distributions by destination entry and weight increment by rate element

Appendix B: Program Lists

Section I: Preparation of IOCS Data

(Program: cadoc00by_rep.f)

```
      Program cadoc00by_rep

      Purpose:  Divides IOCS Clerk and Mailhandler tallies by office group (MODS 1&2, BMCs, Non-MODS),
                activity (mail processing, administrative, window service), and cost pool.

      PLICIT NONE

      integer*4    nfin, npool, npool2

      parameter    (nfin=568)  !  # of MODS finance numbers
      parameter    (npool=57)  !  # of cost pools
      parameter    (npool2=75) !  # of cost pools including BMC and Non-MODS breaks

      include 'iocs2000.h'

      integer*4    pool         !  function to assign cost pool group
      integer*4    rog(nfin)
      integer*4    ct1, ct2, ct3, ctaw1, ctmp1, ctaw2
      integer*4    ctmp2, ctaw3, ctmp3, ctkeep
      integer*4    if262, if260, if257, iw, actv, if244, ct_good
      integer*4    ct_inv, ct_inva, ldc, bmcgrp2
      integer*4    modgrp, nmodgrp, if9805, if9806, if245
      integer*4    keep, hand, bmcgrp, ier, ct, i1
      integer*4    costpool, searchc, i, ct_brk_bmc, ct_brk_nmod
      integer*4    ct_reg_006, ct_reg_after
      integer*4    ct_reg_ldc, ct_reg_60, ct_reg_final
      integer*4    ct_reg_f9606, ct_reg_rpw

      real*8       rf9250, wgt, dlrs, mp_nmod, brk_bmc, brk_nmod
      real*8       mp_mod, mp_bmc, ct_reg_before
      real*8       cost_win, cost_adm, cost_inq, adm_bmc, adm_non
      real*8       win_bmc, win_non, cost_intl, cost_out
      real*8       cost_mod, cost_bmc, cost_nmod
      real*8       ovh6522_bmc, ovh6522_nmod, ovhfact_nmod

      character*6  fin(nfin)
      character*3  type
      character*4  cf244

      ist of MODS 1&2 offices by finance number

      open(10,file='mods_usps.00')

1     format(a6)
      do i=1,nfin
         read(10,11) fin(i)
         rog(i) = 1
      end do

      print*, 'Read in fin #s '

      close(10)

      open(25,file='iocsdata.2000.new',recl=1167) !  FY2000 IOCS data set
1     format(a1167)
1     format(a1167,f15.5,i2,i2,i3,i5)
      open(30,file='mods12_mp00by_new.dat',recl=1200) !  MODS 1&2 mail processing IOCS tallies
      open(35,file='mods12_aw00by_new.dat',recl=1200) !  MODS 1&2 admin/window service IOCS tallies
      open(40,file='bmcs_mp00by_new.dat',recl=1200) !  BMCs mail processing IOCS tallies
      open(45,file='bmcs_aw00by_new.dat',recl=1200) !  BMCs admin/window service IOCS tallies
      open(50,file='nonmods_mp00by_new.dat',recl=1200) !  Non-MODS mail processing IOCS tallies
      open(55,file='nonmods_aw00by_new.dat',recl=1200) !  Non-MODS admwin/window service IOCS tallies
      open(56,file='nonmods_op88_new.dat',recl=1200) !  Non-MODS expedited delivery tallies

      Initialize counter variables

      ier=0
      ct=0
      i1=0
      ct_good = 0
      ct_inv = 0
      ct_inva = 0
       p_nmod = 0.0
      mp_mod = 0.0
      mp_bmc = 0.0
      ct1 = 0
      ct2 = 0
      ct3 = 0
      ctaw1 = 0
```

```
      ctmp1 = 0
      ctaw2 = 0
      ctmp2 = 0
      ctaw3 = 0
      ctmp3 = 0
      ctkeep = 0
       ost_win = 0.0
      cost_adm = 0.0
      cost_inq = 0.0
      cost_intl = 0.0
      cost_out = 0.0
      adm_bmc = 0.0
      adm_non = 0.0
      win_bmc = 0.0
      win_non = 0.0
      ct_brk_bmc = 0
      ct_brk_nmod = 0
      brk_bmc = 0.0
      brk_nmod = 0.0
      cost_mod = 0.0
      cost_bmc = 0.0
      cost_nmod = 0.0
      ovh6522_bmc = 0.0
      ovh6522_nmod = 0.0
      ovhfact_nmod = 0.0
      ct_reg_before = 0.
      ct_reg_006 = 0
      ct_reg_after = 0
      ct_reg_ldc = 0
      ct_reg_60 = 0
      ct_reg_final = 0
      ct_reg_f9606 = 0
      ct_reg_rpw = 0

99    do while (ier.eq.0)

         keep=0
         hand=0
         type=' '
         modgrp=0
         bmcgrp=0
         nmodgrp=0

         read(25,21,iostat=ier,end=100) rec !  Read in IOCS tallies

         iw = 1

         read(f260,'(i2)') if260
         read(f262,'(i4)') if262
         read(f257,'(i2)') if257
         read(f244,'(i4)') if244
         read(f9250,'(f10.0)') rf9250
         read(f9805,'(i4)') if9805
         read(f9806,'(i4)') if9806

         cf244 = f244

         ct=ct+1

c     Separate out Clerk/Mailhander IOCS tallies

c     Identify MODS 1&2 office tallies
         il=searchc(fin,nfin,f2)

c     Identify Clerk/Mailhandler tallies by roster designation (F257)
         if ((if257.eq.11).or.(if257.eq.12).or.(if257.eq.31).or.(if257.eq.32)
     +       .or.(if257.eq.41).or.(if257.eq.42).or.(if257.eq.61).or.(if257.eq.62)
     +       .or.(if257.eq.81).or.(if257.eq.82)) then
            keep=1
         end if

c     Exclude tallies with a tally dollar weight (F9250) of zero
         if (rf9250.le.0.0) then
            keep=0
         end if

c     Exclude CAG K offices
         if (f264.eq.'K') then
            keep=0
         end if
```

```
      if (keep.eq.1) then
         ctkeep=ctkeep+1
      end if

      wgt=rf9250/100000.

:     Assign office type
      if (keep.eq.1) then
         f1(1:1) = '4'        !  Function 4 offices
         if (f2.eq.'      ') then
            f1 = '  '
         end if
         if ((f263.eq.'333333').or.(f263.eq.'444444').or.(f263.eq.'666666')) then
            f1(1:1) = '1'     !  Function 1 offices
         end if
         if (f263.eq.'666666') then !  BMCs
            type = 'bmc'
            ct1 = ct1 + 1
            cost_bmc = cost_bmc + wgt
         else if (i1.gt.0) then !  MODS 1&2 offices
            if ((rog(i1).eq.1).or.(rog(i1).eq.2)) then
               type = 'mod'
               ct2 = ct2 + 1
               cost_mod = cost_mod + wgt
            end if
         else                  !  Non-MODS offices
            type = 'non'
            ct3 = ct3 + 1
            if (if260.ne.88) then
               cost_nmod = cost_nmod + wgt
            end if
         end if

c     Cost pool assignment for MODS 1&2 offices

         if (type.eq.'mod') then

            modgrp=pool(f114) !  Subroutine that assigns cost pool based on MODs code (F114)

            if (modgrp.eq.100) then
               ct_inv = ct_inv + 1
            else
               ct_good = ct_good + 1
            end if

c     Use various IOCS fields to assign cost pool to those tallies with an invalid MODs code

            if (modgrp.eq.100) then
               if (f1(1:1).eq.'1') then !  Function 1 offices
                  if ((f128.eq.'A').and.(f9211.eq.'A')) then
                     modgrp=12 !  manl
                  else if ((f128.eq.'A').and.(f9211.eq.'B')) then
                     modgrp=11 ! manf
                  else if ((f128.eq.'A').and.(f9211.eq.'C')) then
                     modgrp=13 ! manp
                  else if ((f128.eq.'A').and.(f9211.eq.'D')) then
                     modgrp=17 !  1CancMPP
                  else if ((f128.eq.'A').and.(f9211.eq.'E')) then
                     modgrp=16 !  1Bulk pr
                  else if ((f128.eq.'A').and.(f9211.eq.'F')) then
                     modgrp=19 !  1OpPref
                  else if ((f128.eq.'A').and.(f9211.eq.'G')) then
                     modgrp=21 !  1Pouching
                  else if ((f128.eq.'A').and.(f9211.eq.'H')) then
                     modgrp=20 !  1Platform
                  else if (f128.eq.'B') then
                     modgrp=3 !  OCR
                  else if ((f128.ge.'C').and.(f128.le.'E')) then
                     modgrp=1 !  BCS
                  else if (f128.eq.'F') then
                     modgrp=6 !  LSM
                  else if ((f128.ge.'G').and.(f128.le.'H')) then
                     modgrp=17 !  1CancMPP
                  else if (f128.eq.'I') then
                     modgrp=10 !  1SackS_m
                  else if (f128.eq.'J') then
                     modgrp=7 !  mecparc
                  else if (f128.eq.'K') then
                     modgrp=4 !  PSM
```

```
else if (f128.eq.'L') then
   modgrp=8 !  spbs Oth
else if (f128.eq.'M') then
   modgrp=10 !  1Sacks_m
else if (f128.eq.'N') then
   modgrp=22 !  1Sacks_h
else if (f128.eq.'O') then
   modgrp=19 !  1OPref
else if (f128.eq.'P') then
   modgrp=23 !  1Scan
else if (f128.eq.'Q') then
   modgrp=21 !  1Pouching
else if (f128.eq.'R') then
   modgrp=17 !  1CancMPP
else if (f128.eq.'S') then
   modgrp=15 !  LDC 15
else if ((f128.eq.'T').and.((f9212.ge.'A').and.(f9212.le.'D'))) then !
   modgrp=20 !  1Platform
else if (if260.eq.7) then
   modgrp=42 !  LDC 79
else if (if260.eq.8) then
   modgrp=20 !  1Platform
else if ((f118.eq.'A').or.(f118.eq.'C').or.
+       (f118.eq.'E').or.(f118.eq.'F').or.
+       (f118.eq.'I').or.(f118.eq.'K')) then
   modgrp=17 !  1CancMPP
else if ((f118.eq.'B').or.(f118.eq.'D').or.
+       (f118.eq.'H').or.(f118.eq.'J')) then
   modgrp=19 !  1OPref
else if (f118.eq.'G') then
   modgrp=28 !  Rewrap
else if (((f119.ge.'A').and.(f119.le.'F')).and.
+       (f122.eq.' ').and.(f9602.eq.'A')) then
   modgrp=13 !  manp
else if (((f119.ge.'A').and.(f119.le.'F')).and.
+       (f122.eq.' ').and.(f9602.eq.'C')) then
   modgrp=22 !  1Sacks_h
else if (((f119.ge.'A').and.(f119.le.'F')).and.
+       (f122.eq.' ').and.(f9602.eq.'D')) then
   modgrp=13 !  manp
else if (((f119.ge.'A').and.(f119.le.'G')).and.
       (f122.eq.' ').and.((f128.eq.'A').and.(f9211.eq.'I'))) then !
   modgrp=30 !  1Misc
else if (((f119.ge.'A').and.(f119.le.'G')).and.
+       (f122.eq.' ').and.(f9602.eq.'B')) then
   modgrp=21 !  1Pouching
else if ((f122.ge.'A').and.(f122.le.'F')) then
   modgrp=21 !  1Pouching
else if ((f122.ge.'I').and.(f122.le.'L')) then
   modgrp=21 !  1Pouching
else if (f122.eq.'G') then
   modgrp=30 !  1Misc
else if (f122.eq.'M') then
   modgrp=30 !  1Misc
else if (f122.eq.'H') then
   modgrp=29 !  1EEqmt
else if (if260.eq.0) then
   modgrp=24 !  Bus Reply
else if (if260.eq.6) then
   modgrp=31 !  1Support
else if (if260.eq.9) then
   modgrp=95 !  1Window
else if (if260.eq.10) then
   modgrp=98 !  2Adm
else if (if260.eq.14) then
   modgrp=41 !  LDC 49
else if (if260.eq.17) then
   modgrp=97 !  2Adm inq
else if (if260.eq.18) then
   modgrp=27 !  Registry
else if (if260.eq.19) then
   modgrp=26 !  Mailgram
else if (if260.eq.20) then
   modgrp=31 !  1Support
else if (if260.eq.21) then
   modgrp=38 !  LDC48 Oth
else if (if260.eq.22) then
   modgrp=25 !  Express
else if (if260.eq.23) then
   modgrp=38 !  LDC48 Oth
```

```
            else if ((if260.ge.24).and.(if260.le.26)) then
               modgrp=95 !  Window
            else
               modgrp=30 !  1Misc
            end if
         end if

         if (f1(1:1).eq.'4') then !  Function 4 offices
            modgrp = 98 !  2Adm
            if ((f128.ge.'B').and.(f128.le.'E')) then
               modgrp=33 !  LDC 41
            else if ((f128.eq.'F').or.(f128.eq.'K')) then
               modgrp=34 !  LDC 42
            else if (if260.eq.0) then
               modgrp=40 !  LDC48 Sp Serv
            else if (if260.eq.6) then
               modgrp=40 !  LDC48 Sp Serv
            else if ((if260.ge.11).and.(if260.le.13).or.
     +           (if260.eq.20)) then
               modgrp=36 !  LDC 44
            else if ((if260.eq.9).or.((if260.ge.24).and.
     +           (if260.le.26))) then
               modgrp=95 !  Window
            else if (if260.eq.10) then
               modgrp=98 !  2Adm
            else if (if260.eq.14) then
               modgrp=41 !  LDC 49
            else if (if260.eq.17) then
               modgrp=97 !  2Adm inq
            else if (if260.eq.18) then
               modgrp=40 !  LDC48 Sp Serv
            else if (if260.eq.19) then
               modgrp=26 !  Mailgram
            else if (if260.eq.21) then
               modgrp=40 !  LDC 48 Sp Serv
            else if (if260.eq.22) then
               modgrp=37 !  LDC48 Exp
            else if (if260.eq.23) then
               modgrp=40 !  LDC48 Sp Serv
            else
               modgrp=35 !  LDC43
            end if
         end if
      end if

      if (modgrp.eq.100) then
         ct_inva = ct_inva + 1
         print*, 'Cost pool not assigned ', f114
      end if

c     Assigns LDC to cost pools

      ldc = 0

      if ((modgrp.ge.1).and.(modgrp.le.3)) then
         ldc = 11
      else if ((modgrp.ge.4).and.(modgrp.le.6)) then
         ldc = 12
      else if ((modgrp.ge.7).and.(modgrp.le.10)) then
         ldc = 13
      else if ((modgrp.ge.11).and.(modgrp.le.14)) then
         ldc = 14
      else if (modgrp.eq.15) then
         ldc = 15
      else if ((modgrp.ge.16).and.(modgrp.le.23)) then
         ldc = 17
      else if ((modgrp.ge.24).and.(modgrp.le.31)) then
         ldc = 18
      else if (modgrp.eq.33) then
         ldc = 41
      else if (modgrp.eq.34) then
         ldc = 42
      else if (modgrp.eq.35) then
         ldc = 43
      else if (modgrp.eq.36) then
         ldc = 44
      else if ((modgrp.ge.37).and.(modgrp.le.40)) then
         ldc = 48
      else if (modgrp.eq.41) then
         ldc = 49
```

```
            if (actv.eq.1060) actv=1068
            if (actv.eq.2060) actv=2068
            if (actv.eq.3060) actv=3068
            if (actv.eq.4060) actv=4068
         end if
      end if
      if ((actv.eq.1310).or.(actv.eq.2310).or.(actv.eq.3310).or.(actv.eq.4310)) then !  Reg ECR
         if ((f9618.eq.'1').or.(f9619.eq.'1')) then !  WSH/WSS
            if (actv.eq.1310) actv=1311
            if (actv.eq.2310) actv=2311
            if (actv.eq.3310) actv=3311
            if (actv.eq.4310) actv=4311
         else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
            if (actv.eq.1310) actv=1312
            if (actv.eq.2310) actv=2312
            if (actv.eq.3310) actv=3312
            if (actv.eq.4310) actv=4312
         else if (f9617.eq.'1') then !  ECRLOT
            actv = actv
         else
            actv = actv
         end if
      end if
      if ((actv.eq.1330).or.(actv.eq.2330).or.(actv.eq.3330).or.(actv.eq.4330)) then !  NP ECR
         if ((f9618.eq.'1').or.(f9619.eq.'1')) then !  WSH/WSS
            if (actv.eq.1330) actv=1331
            if (actv.eq.2330) actv=2331
            if (actv.eq.3330) actv=3331
            if (actv.eq.4330) actv=4331
         else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
            if (actv.eq.1330) actv=1332
            if (actv.eq.2330) actv=2332
            if (actv.eq.3330) actv=3332
            if (actv.eq.4330) actv=4332
         else if (f9617.eq.'1') then !  ECRLOT
            actv = actv
         else
            actv = actv
         end if
      end if

      if (if260.eq.0) then
         if260=30
      end if

c     Assigns LDC to cost pools

      if (nmodgrp.eq.50) then
         ldc = 17
      else if (nmodgrp.eq.51) then
         ldc = 11
      else if ((nmodgrp.eq.52).or.((nmodgrp.ge.56).and.(nmodgrp.le.57))) then
         ldc = 18
      else if ((nmodgrp.ge.53).and.(nmodgrp.le.55)) then
         ldc = 14
      else
         ldc = 0
      end if

      if (((if260.ge.9).and.(if260.le.10)).or.
     +    (if260.eq.17).or.((if260.ge.24).and.
     +    (if260.le.26))) then !  Admin/Window Service
         if ((if260.eq.9).or.((if260.ge.24).and.(if260.le.26))) then
            costpool = 1
         else
            costpool = 2
         end if
         dlrs=wgt * 4833550./4401822. !  Convert to cost pool dollars
         write(55,31) rec, dlrs, if260, costpool, iw, actv ! Admin/Window Service tallies
         adm_non = adm_non + wgt
         ctaw3=ctaw3+1
         if ((if260.eq.9).or.((if260.ge.24).and.(if260.le.26))) then !  Window Service
            win_non = win_non + wgt
         end if
         if (actv.ne.6522) then
            ovh6522_nmod = ovh6522_nmod + (wgt*4833550./4401822.) !  Overhead factor
         end if
      else if (if260.ne.88) then !  exclude expedited delivery
         dlrs=wgt
         if (nmodgrp.gt.0) then
```

```
         else if (modgrp.eq.42) then
            ldc = 79
         else
            ldc = 0
         end if

:    MODS-based encirclement rule

:    For domestic mail
         if (modgrp.eq.32) then !  Intl
            if ((f9805(1:2).eq.'54').or.(((f9805(2:2).ge.'6').and.
     &         (f9805(2:2).le.'8')).and.(if9805.le.4950))) then
               ct_reg_before = ct_reg_before + wgt
            end if
         end if

         if (((f245.ge.'001').and.(f245.le.'030')).or.
     &      ((f246.ge.'001').and.(f246.le.'030')).or.
     &      ((if9806.ge.10).and.(if9806.le.300))) then
            if ((f245.eq.'006').or.(f246.eq.'006')) then
               actv = 60
            else if (f245.eq.'019') then
               if ((f9606.eq.'A').and.(f9606.eq.'B')) then
                  actv = 190
               else if ((f245.eq.'019').and.(f9805(2:4).eq.'510')) then
                  actv = 190
               else if (((f246.eq.'   ').or.(f247.eq.'   ').or.
     &                (f248.eq.'   ').or.(f249.eq.'   ')).and.
     &                ((modgrp.eq.42).or.(modgrp.eq.95).or.(modgrp.eq.35).or. !  LD79, Window, LD43
     &                (modgrp.eq.38).or.(modgrp.eq.40).or.(modgrp.eq.39).or. !  LD48 Oth, LD48 SSV, LD48 Adm
     &                (modgrp.ge.97).or.(modgrp.eq.30).or.(modgrp.eq.31))) then !  2Adm, 1Misc, 1Support
                  actv = 190
               else
                  actv = if9805
               end if
            else if ((f245.eq.'030').and.((f9606.eq.'C').or.(f9632.eq.'1'))) then
               actv = 300
            else if ((f246.eq.'   ').and.(f247.eq.'   ').and.
     &           (f248.eq.'   ').and.(f249.eq.'   ')) then
               actv = if9805
               if (f245.eq.'021') then
                  actv = 210
               else if ((f245.eq.'009').and.
     &                ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
     &                (modgrp.eq.16).or.(modgrp.eq.23).or.(modgrp.eq.21).or. !  1Bulk pr, 1Scan, 1Pouching
     &                (modgrp.eq.17).or.(modgrp.eq.19).or.(modgrp.eq.18).or. !  1CancMPP, 1OpPref, 1OpBulk
     &                (modgrp.eq.22).or.(modgrp.eq.30).or.(modgrp.eq.31).or. !  1Sacks_h, 1Misc, 1Support
     &                (modgrp.eq.35).or.(modgrp.eq.39).or.(modgrp.ge.97))) then !  LD43, LD48 Adm, 2Adm
                  actv = 90
               else if (((f245.eq.'003').or.(f245.eq.'007').or.(f245.eq.'008')).and.
     &                ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
     &                (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &                (modgrp.eq.95).or. !  Window
     &                (modgrp.eq.97).or.(modgrp.eq.30).or.(modgrp.eq.31).or. !  2Adm Inq, 1Misc, 1Support
     &                (modgrp.eq.39).or.(modgrp.ge.97))) then !  LD48 Adm, 2Adm
                  read(f245,'(i3)') if245
                  actv = if245*10
               else if ((f245.eq.'001').and.
     &                ((modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &                (modgrp.eq.95).or. !  Window
     &                (modgrp.eq.38).or.(modgrp.eq.40).or.(modgrp.eq.37).or. !  LD48 Oth, LD48 SSV, LD48 Exp
     &                (modgrp.eq.25).or.(modgrp.eq.30).or.(modgrp.eq.31))) then !  Express, 1Misc, 1Support
                  actv = 10
               else if ((f245.eq.'005').and.
     &                ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
     &                (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &                (modgrp.eq.95).or. !  Window
     &                (modgrp.eq.30).or.(modgrp.eq.31).or. !  1Misc, 1Support
     &                (modgrp.eq.39).or.(modgrp.ge.97))) then !  LD48 Adm, 2Adm
                  actv = 50
               else if ((f245.eq.'002').and.
     &                ((modgrp.eq.16).or.(modgrp.eq.23).or.(modgrp.eq.21).or. !  1Bulk pr, 1Scan, 1Pouching
     &                (modgrp.eq.17).or.(modgrp.eq.19).or.(modgrp.eq.18).or. !  1CancMPP, 1OpPref, 1OpBulk
     &                (modgrp.eq.22).or. !  1SackS_h
     &                (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &                (modgrp.eq.95))) then !  Window
                  actv = 20
               end if
            else if ((f246.gt.'001').or.(f247.gt.'001').or.
     &           (f248.gt.'001').or.(f249.gt.'001')) then
```

```
            actv = if9805
            if (((f245.eq.'003').or.(f245.eq.'007').or.
     &          (f245.eq.'008')).and.
     &          ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
     &          (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
                (modgrp.eq.95).or. !  Window
                (modgrp.eq.97).or.(modgrp.eq.30).or.(modgrp.eq.31).or. !  2Adm Inq, 1Misc, 1Support
     &          (modgrp.eq.39).or.(modgrp.eq.ge.97))) then !  LD48 Adm, 2Adm
               read(f245,'(i3)') if245
               actv = if245*10
            else if ((f245.eq.'001').and.
     &               ((modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &               (modgrp.eq.95).or. !  Window
     &               (modgrp.eq.38).or.(modgrp.eq.40).or.(modgrp.eq.37).or. !  LD48 Oth, LD48 SSV, LD48 Exp
     &               (modgrp.eq.25).or.(modgrp.eq.30).or.(modgrp.eq.31))) then !  Express, 1Misc, 1Support
               actv = 10
            else if ((f245.eq.'005').and.
     &               ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
     &               (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &               (modgrp.eq.95).or. !  Window
     &               (modgrp.eq.30).or.(modgrp.eq.31).or. !  1Misc, 1Support
     &               (modgrp.eq.39).or.(modgrp.eq.97))) then !  LD48 Adm, 2Adm
               actv = 50
            else if ((f245.eq.'002').and.
     &               ((modgrp.eq.16).or.(modgrp.eq.23).or.(modgrp.eq.21).or. !  1Bulk pr, 1Scan, 1Pouching
     &               (modgrp.eq.17).or.(modgrp.eq.19).or.(modgrp.eq.18).or. !  1CancMPP, 1OpPref, 1OpBulk
     &               (modgrp.eq.22).or. !  1SackS_h
     &               (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &               (modgrp.eq.95))) then !  Window
               actv = 20
            end if
         end if
      else
         actv = if9806
      end if

c     For international mail
      if ((f9805(1:2).eq.'54').or.(((f9805(2:2).ge.'6').and.
     &   (f9805(2:2).le.'8')).and.(if9805.le.4950))) then
         if (((f245.ge.'001').and.(f245.le.'030')).or.
            ((f246.ge.'001').and.(f246.le.'030'))) then
            if ((f245.eq.'006').and.(f246.eq.'006')) then
               actv = 700
            else if (f245.eq.'019') then
               if ((f9606.eq.'A').or.(f9606.eq.'B')) then
                  actv = 700
               else if ((f245.eq.'019').and.(f9805(2:4).eq.'510')) then
                  actv = 700
               else if (((f246.eq.'   ').or.(f247.eq.'   ').or.
     &                   (f248.eq.'   ').or.(f249.eq.'   ')).and.
     &                   ((modgrp.eq.42).or.(modgrp.eq.95).or.(modgrp.eq.35).or. !  LD79, Window, LD43
     &                   (modgrp.eq.38).or.(modgrp.eq.40).or.(modgrp.eq.39).or. !  LD48 Oth, LD48 SSV, LD48 Adm
     &                   (modgrp.ge.97).or.(modgrp.eq.30).or. !  2Adm, 1Misc
     &                   (modgrp.eq.31))) then !  1Support
                  actv = 700
               else
                  actv = if9805
               end if
            else if ((f245.eq.'030').and.((f9606.eq.'C').or.(f9632.eq.'1'))) then
               actv = 700
            else if ((f246.eq.'   ').and.(f247.eq.'   ').and.
     &                (f248.eq.'   ').and.(f249.eq.'   ')) then
               actv = if9805
               if (f245.eq.'021') then
                  actv = 700
               else if ((f245.eq.'009').and.
     &                   ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
     &                   (modgrp.eq.16).or.(modgrp.eq.23).or.(modgrp.eq.21).or. !  1Bulk pr, 1Scan, 1Pouching
     &                   (modgrp.eq.17).or.(modgrp.eq.19).or.(modgrp.eq.18).or. !  1CancMPP, 1OpPref, 1OpBulk
     &                   (modgrp.eq.22).or.(modgrp.eq.30).or.(modgrp.eq.31).or. !  1SackS_h, 1Misc, 1Support
     &                   (modgrp.eq.35).or.(modgrp.eq.39).or.(modgrp.ge.97))) then !  LD43, LD48 Adm, 2Adm
                  actv = 700
               else if (((f245.eq.'003').or.(f245.eq.'007').or.(f45.eq.'008')).and.
                        ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
                        (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
     &                  (modgrp.eq.95).or. !  Window
     &                  (modgrp.eq.97).or.(modgrp.eq.30).or.(modgrp.eq.31).or. !  2Adm Inq, 1Misc, 1Support
     &                  (modgrp.eq.39).or.(modgrp.ge.97))) then !  LD48 Adm, 2Adm
                  actv = 700
               else if ((f245.eq.'001').and.
```

```
&                    ((modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
&                    (modgrp.eq.95).or. !  Window
&                    (modgrp.eq.38).or.(modgrp.eq.40).or.(modgrp.eq.37).or. !  LD48 Oth, LD48 SSV, LD48 Exp
&                    (modgrp.eq.25).or.(modgrp.eq.30).or.(modgrp.eq.31))) then !  Express, 1Misc, 1Support
                  actv = 700
               else if ((f245.eq.'005').and.
                    ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
&                   (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
&                   (modgrp.eq.95).or. !  Window
&                   (modgrp.eq.30).or.(modgrp.eq.31).or. !  1Misc, 1Support
&                   (modgrp.eq.39).or.(modgrp.ge.97))) then !  LD48 Adm, 2Adm
                  actv = 700
               else if ((f245.eq.'002').and.
&                   ((modgrp.eq.16).or.(modgrp.eq.23).or.(modgrp.eq.21).or. !  1Bulk pr, 1Scan, 1Pouching
&                   (modgrp.eq.17).or.(modgrp.eq.19).or.(modgrp.eq.18).or. !  1CancMPP, 1OpPref, 1OpBulk
&                   (modgrp.eq.22).or. !  1SackS_h
&                   (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
&                   (modgrp.eq.95))) then !  Window
                  actv = 700
               end if
            else if ((f246.gt.'001').or.(f247.gt.'001').or.
&                   (f248.gt.'001').or.(f249.gt.'001')) then
               actv = if9805
               if (((f245.eq.'003').or.(f245.eq.'007').or.(f245.eq.'008')).and.
&                  ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
&                  (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
&                  (modgrp.eq.95).or. !  Window
&                  (modgrp.eq.97).or.(modgrp.eq.30).or.(modgrp.eq.31).or. !  2Adm Inq, 1Misc, 1Support
&                  (modgrp.eq.39).or.(modgrp.ge.97))) then !  LD48 Adm, 2Adm
                  actv = 700
               else if ((f245.eq.'001').and.
&                   ((modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
&                   (modgrp.eq.95).or. !  Window
&                   (modgrp.eq.38).or.(modgrp.eq.40).or.(modgrp.eq.37).or. !  LD48 Oth, LD48 SSV, LD48 Exp
&                   (modgrp.eq.25).or.(modgrp.eq.30).or.(modgrp.eq.31))) then !  Express, 1Misc, 1Support
                  actv = 700
               else if ((f245.eq.'005').and.
&                   ((modgrp.eq.24).or.(modgrp.eq.38).or.(modgrp.eq.40).or. !  BusReply, LD48 Oth, LD48 SSV
&                   (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
                    (modgrp.eq.95).or. !  Window
                    (modgrp.eq.30).or.(modgrp.eq.31).or. !  1Misc, 1Support
&                   (modgrp.eq.39).or.(modgrp.ge.97))) then !  LD48 Adm, 2Adm
                  actv = 700
               else if ((f245.eq.'002').and.
&                   ((modgrp.eq.16).or.(modgrp.eq.23).or.(modgrp.eq.21).or. !  1Bulk pr, 1Scan, 1Pouching
&                   (modgrp.eq.17).or.(modgrp.eq.19).or.(modgrp.eq.18).or. !  1CancMPP, 1OpPref, 1OpBulk
&                   (modgrp.eq.22).or. !  1SackS_h
&                   (modgrp.eq.42).or.(modgrp.eq.20).or.(modgrp.eq.35).or. !  LD79, 1Platform, LD43
&                   (modgrp.eq.95))) then !  Window
                  actv = 700
               end if
            end if
         end if
      end if

      if (modgrp.eq.27) then !  Registry
         if (actv.eq.60) then
            ct_reg_after = ct_reg_after + 1
         end if
      end if

      if (((ldc.ge.11).and.(ldc.le.17)).or.
&        ((ldc.ge.41).and.(ldc.le.44)).or.(ldc.eq.79)) then
         actv = if9805
      end if

      if (modgrp.eq.27) then !  Registry
         if (actv.eq.60) then
            ct_reg_ldc = ct_reg_ldc + 1
         end if
      end if

      if ((modgrp.eq.24).and.(actv.ne.90)) then !  BusReply
         actv = if9805
      end if

      if ((modgrp.eq.27).and.(actv.ne.60)) then !  Registry
         actv = if9805
      end if
```

```
            if ((modgrp.eq.41).and.(actv.lt.100)) then !  LD49
               actv = if9805
            end if

            if (modgrp.eq.27) then !  Registry
               if (actv.eq.60) then
                  ct_reg_60 = ct_reg_60 + 1
               end if
            end if

         Special service operations in International
            if (if9806.eq.700) then !  Intl Sp Serv
               if ((f114.eq.'578').or.(f114.eq.'580').or.
     &             (f114.eq.'681').or.(f114.eq.'573').or.
     &             (f114.eq.'577')) then
                  if ((f245.ge.'001').and.(f245.le.'030')) then
                     actv = 700
                  else
                     actv = if9805
                  end if
               else if ((ldc.eq.18).or.(ldc.eq.48)) then
                  if ((f2.eq.'054521').or.(f2.eq.'054522').or.
     &                (f2.eq.'056793').or.(f2.eq.'160049').or.
     &                (f2.eq.'115855').or.(f2.eq.'350185').or.
     &                (f2.eq.'482267').or.(f2.eq.'054520').or.
     &                (f2.eq.'054523').or.(f2.eq.'055509').or.
     &                (f2.eq.'055513').or.(f2.eq.'056790').or.
     &                (f2.eq.'115851').or.(f2.eq.'115852').or.
     &                (f2.eq.'160046').or.(f2.eq.'160047').or.
     &                (f2.eq.'252493').or.(f2.eq.'268363').or.
     &                (f2.eq.'333869').or.(f2.eq.'350185').or.
     &                (f2.eq.'351029').or.(f2.eq.'482267').or.
     &                (f2.eq.'482268').or.(f2.eq.'484313').or.
     &                (f2.eq.'512704').or.(f2.eq.'512705').or.
     &                (f2.eq.'547619')) then
                     actv = 700
                  else
                     actv = if9805
                  end if
               else
                  actv = if9805
               end if
            end if

c     Special handling that will incur special serv costs in any pool
c     has to be associated with Std B or Std A Single Piece
            if (((if9806.eq.20).or.(f245.eq.'002').or.(f246.eq.'002').or.
     &           (f247.eq.'002').or.(f248.eq.'002').or.(f249.eq.'002')).and.
     &           ((if9805.ge.1000).and.(if9805.le.4950))) then
               if ((f9805(1:3).eq.'360').or.(f9805(2:2).eq.'4')) then
                  actv = 20
               else
                  actv = if9805
               end if
            end if

c        Detached forms that will incur special service costs in any pool
            if (f9635.eq.'C') then !  USPS form shape
               actv = actv
               if ((f9606.ge.'A').and.(f9606.le.'B')) then
                  actv = 190
               end if
               if ((f9632.eq.'1').and.(f9606.eq.'C')) then
                  actv = 300
               end if
               if (f9606.eq.'D') actv = if9805
               if (f9606.eq.'E') actv = 100
               if (f9606.eq.'F') actv = 60
               if (f9606.eq.'G') actv = if9805
               if (f9606.eq.'H') actv = 60
               if (f9606.eq.'I') actv = if9805
            end if

            if (modgrp.eq.27) then
               if (actv.eq.60) then
                  ct_reg_f9606 = ct_reg_f9606 + 1
               end if
            end if

c     Adjustment to be consistent with what's included in RPW pieces
```

```
            if (actv.eq.60) then
               if ((f9805(2:2).eq.'8').and.((if9805.ge.1000).and.
     &            (if9805.le.4950))) then
                  actv = if9805
               else if ((f9805(2:4).eq.'510').and.((if9805.ge.1000).and.
                     (if9805.le.4950))) then
                  actv = if9805
               else if ((f9805(1:3).ge.'545').and.(f9805(1:3).le.'548')) then
                  actv = if9805
               else
                  actv = 60
               end if
            end if

            if (modgrp.eq.27) then !  Registry
               if (actv.eq.60) then
                  ct_reg_rpw = ct_reg_rpw + 1
               end if
            end if

c     Correction for business reply - incl BRMAS
            if ((f245.eq.'009').or.(f246.eq.'009')) then
               if ((modgrp.eq.24).or.(f120.eq.'F')) then
                  actv = 90
               else
                  actv = if9805
               end if
            end if

            if ((modgrp.ge.97).or.(modgrp.eq.95)) then !  Admin/Window Service cost pools
               actv = if9806
            end if

            if ((if9806.eq.110).or.(if9806.eq.120)) then
               actv = if9806
               print *,'Del conf tally; pool =',modgrp,' $=',wgt
            end if

c     Form Intl/MP cost pool
            if ((f2.eq.'054521').or.(f2.eq.'054522').or.(f2.eq.'056793').or.
               (f2.eq.'160049').or.(f2.eq.'115855').or.(f2.eq.'350185').or.
     &         (f2.eq.'482267')) then
               if (((ldc.ge.11).and.(ldc.le.18)).or.
     &            ((ldc.ge.41).and.(ldc.le.44)).or.
     &            ((ldc.ge.48).and.(ldc.le.49)).or.
     &            (ldc.eq.79)) then
                  modgrp = 32 !  Intl MP
                  ldc = 19
               else
                  modgrp = 96 !  Intl Admin
               end if
            end if

            if (modgrp.eq.27) then !  Registry
               if (actv.eq.60) then
                  ct_reg_final = ct_reg_final + 1
               end if
            end if

c  Reassign specific actv codes for expanded subclasses

            if ((actv.eq.1060).or.(actv.eq.2060).or.(actv.eq.3060).or.(actv.eq.4060)) then !  1st SP
               if (f136.eq.'D') then !  Metered
                  if (actv.eq.1060) actv=1068
                  if (actv.eq.2060) actv=2068
                  if (actv.eq.3060) actv=3068
                  if (actv.eq.4060) actv=4068
               end if
            end if
            if ((actv.eq.1310).or.(actv.eq.2310).or.(actv.eq.3310).or.(actv.eq.4310)) then !  Reg ECR
               if ((f9618.eq.'1').or.(f9619.eq.'1')) then !  WSH/WSS
                  if (actv.eq.1310) actv=1311
                  if (actv.eq.2310) actv=2311
                  if (actv.eq.3310) actv=3311
                  if (actv.eq.4310) actv=4311
               else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
                  if (actv.eq.1310) actv=1312
                  if (actv.eq.2310) actv=2312
                  if (actv.eq.3310) actv=3312
                  if (actv.eq.4310) actv=4312
```

```
            else if (f9617.eq.'1') then !  ECRLOT
               actv = actv
            else
               actv = actv
            end if
         end if
         if ((actv.eq.1330).or.(actv.eq.2330).or.(actv.eq.3330).or.(actv.eq.4330)) then !  NP ECR
            if ((f9618.eq.'1').or.(f9619.eq.'1')) then !  WSH/WSS
               if (actv.eq.1330) actv=1331
               if (actv.eq.2330) actv=2331
               if (actv.eq.3330) actv=3331
               if (actv.eq.4330) actv=4331
            else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
               if (actv.eq.1330) actv=1332
               if (actv.eq.2330) actv=2332
               if (actv.eq.3330) actv=3332
               if (actv.eq.4330) actv=4332
            else if (f9617.eq.'1') then !  ECRLOT
               actv = actv
            else
               actv = actv
            end if
         end if

         if (modgrp.ge.95) then !  Admin/Window Service cost pools
            if (modgrp.eq.95) then !  Window Service
               cost_win = cost_win + wgt
            else if (modgrp.eq.99) then !  2Adm out
               cost_out = cost_out + wgt
               dlrs=0.0
            else if (modgrp.eq.96) then !  2Adm intl
               cost_intl = cost_intl + wgt
            else
               if (modgrp.eq.97) then !  2Adm inq  (claims/inquiry)
                  cost_inq = cost_inq + wgt
               else if (modgrp.eq.98) then !  2Adm
                  cost_adm = cost_adm + wgt
               end if
            end if
            if (modgrp.eq.95) then !  Window Service
               costpool=1
            else
               costpool=2
            end if
            if ((modgrp.ge.95).and.(modgrp.le.99)) then
               write(35,31) rec, dlrs, modgrp, costpool, iw, actv !  Admin/Window Service tallies
               ctaw2=ctaw2+1
            end if
         else
            write(30,31) rec, wgt, modgrp, ldc, iw, actv !  Mail Proc tallies
            ctmp2=ctmp2+1
            mp_mod = mp_mod + wgt
         end if

      end if              ! Tallies at MODS offices

c     Cost pool assignment for BMCs

      if (type.eq.'bmc') then
         if (((if260.ge.0).and.(if260.le.8)).or.
     +          ((if260.ge.11).and.(if260.le.16)).or.
     +          ((if260.ge.18).and.(if260.le.23)).or.
     +          ((if260.ge.27).and.(if260.le.29)).or.(if260.eq.88)) then !  Mail proc operation codes (P260)
            if (if9806.eq.6521) then
               bmcgrp = 75 ! Z_breaks
            else if ((f128.eq.'I').and.(f121.eq.'N')) then
               bmcgrp=47  ! SSM
            else if ((f128.eq.'I').and.(f121.eq.'Y')) then
               bmcgrp=45  ! SSM_Alli
            else if ((f128.eq.'J').and.(f121.eq.'N')) then
               bmcgrp=46  ! PSM
            else if ((f128.eq.'J').and.(f121.eq.'Y')) then
               bmcgrp=45  ! PSM_Alli
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
                  (f128.eq.'M')) then
               bmcgrp=49  ! NMO
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +             (f9211.eq.'C').and.(f9602.eq.'C')) then
               bmcgrp=49  ! NMO
            else if (((f119.ge.'A').and.(f119.le.'G')).and.
```

```
     +                (f9211.eq.'C').and.(f9602.eq.'D')) then
                   bmcgrp=49  ! NMO
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f128.eq.'L')) then
                   bmcgrp=48  ! SPB
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f9602.eq.'A')) then
                   bmcgrp=48  ! SPB
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f9602.eq.'B')) then
                   bmcgrp=48  ! SPB
                else if (((f116.ge.'A').and.(f116.le.'H')).and.
     +                (f9209.eq.' ')) then
                   bmcgrp=44  ! Platform
                else if (((f118.ge.'A').and.(f118.le.'K')).and.
     +                (f9209.eq.' ')) then
                   bmcgrp=45  ! Mail Prep
                else
                   bmcgrp=45  ! Other
                end if
                if ((f128.eq.'I').and.(f121.eq.'N')) then
                   bmcgrp2=47 ! SSM
                else if ((f128.eq.'I').and.(f121.eq.'Y')) then
                   bmcgrp2=45 ! SSM_Alli
                else if ((f128.eq.'J').and.(f121.eq.'N')) then
                   bmcgrp2=46 ! PSM
                else if ((f128.eq.'J').and.(f121.eq.'Y')) then
                   bmcgrp2=45 ! PSM_Alli
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f128.eq.'M')) then
                   bmcgrp2=49 ! NMO
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f9211.eq.'C').and.(f9602.eq.'C')) then
                   bmcgrp2=49 ! NMO
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f9211.eq.'C').and.(f9602.eq.'D')) then
                   bmcgrp2=49 ! NMO
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
                         (f128.eq.'L')) then
                   bmcgrp2=48 ! SPB
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f9602.eq.'A')) then
                   bmcgrp2=48 ! SPB
                else if (((f119.ge.'A').and.(f119.le.'G')).and.
     +                (f9602.eq.'B')) then
                   bmcgrp2=48 ! SPB
                else if (((f116.ge.'A').and.(f116.le.'H')).and.
     +                (f9209.eq.' ')) then
                   bmcgrp2=44 ! Platform
                else if (((f118.ge.'A').and.(f118.le.'K')).and.
     +                (f9209.eq.' ')) then
                   bmcgrp2=45 ! Mail Prep
                else
                   bmcgrp2=45 ! Other
                end if
             else
                bmcgrp=0
             end if

             actv = if9806

c     BMC encirclements

             if (((bmcgrp.ge.44).and.(bmcgrp.le.49)).and.(actv.eq.60)) then
                if ((f9805(2:4).eq.'510').and.((if9805.ge.1000).and.(if9805.le.4950))) then
                   actv = if9805
                else if ((f9805(2:2).eq.'8').and.
     &                ((if9805.ge.1000).and.(if9805.le.4950))) then
                   actv = if9805
                else if ((f9805(1:3).ge.'545').and.(f9805(1:3).le.'548')) then
                   actv = if9805
                else
                   actv = 60
                end if
             end if

c  Reassign specific actv codes for expanded subclasses

             if ((actv.eq.1060).or.(actv.eq.2060).or.(actv.eq.3060).or.(actv.eq.4060)) then !  1st SP
                if (f136.eq.'D') then !  Metered
```

```
            if (actv.eq.1060) actv=1068
            if (actv.eq.2060) actv=2068
            if (actv.eq.3060) actv=3068
            if (actv.eq.4060) actv=4068
         end if
      end if
      if ((actv.eq.1310).or.(actv.eq.2310).or.(actv.eq.3310).or.(actv.eq.4310)) then !  Reg ECR
         if ((f9618.eq.'1').or.(f9619.eq.'1')) then !  WSH/WSS
            if (actv.eq.1310) actv=1311
            if (actv.eq.2310) actv=2311
            if (actv.eq.3310) actv=3311
            if (actv.eq.4310) actv=4311
         else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
            if (actv.eq.1310) actv=1312
            if (actv.eq.2310) actv=2312
            if (actv.eq.3310) actv=3312
            if (actv.eq.4310) actv=4312
         else if (f9617.eq.'1') then !  ECRLOT
            actv = actv
         else
            actv = actv
         end if
      end if
      if ((actv.eq.1330).or.(actv.eq.2330).or.(actv.eq.3330).or.(actv.eq.4330)) then !  NP ECR
         if ((f9618.eq.'1').or.(f9619.eq.'1')) then !  WSH/WSS
            if (actv.eq.1330) actv=1331
            if (actv.eq.2330) actv=2331
            if (actv.eq.3330) actv=3331
            if (actv.eq.4330) actv=4331
         else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
            if (actv.eq.1330) actv=1332
            if (actv.eq.2330) actv=2332
            if (actv.eq.3330) actv=3332
            if (actv.eq.4330) actv=4332
         else if (f9617.eq.'1') then !  ECRLOT
            actv = actv
         else
            actv = actv
         end if
      end if

c     Assigns LDC to cost pools

      if ((bmcgrp.ge.46).and.(bmcgrp.le.48)) then
         ldc = 13
      else if (bmcgrp.eq.49) then
         ldc = 14
      else if ((bmcgrp.ge.44).and.(bmcgrp.le.45)) then
         ldc = 17
      else
         ldc = 0
      end if

      if (bmcgrp.eq.0) then
         if ((if260.eq.9).or.((if260.ge.24).and.(if260.le.26))) then !  Admin/Window Service
            costpool = 1
         else
            costpool = 2
         end if
         dlrs=wgt * 850133./849454. !  Convert to cost pool dollars
         write(45,31) rec, dlrs, bmcgrp, costpool, iw, actv !  Admin/Window Service tallies
         ctaw1=ctaw1+1
         adm_bmc = adm_bmc + wgt
         if ((if260.eq.9).or.((if260.ge.24).and.(if260.le.26))) then !  Window Service
            win_bmc = win_bmc + wgt
         end if
         if (actv.ne.6522) then
            ovh6522_bmc = ovh6522_bmc + (wgt*850133./849454.) !  Overhead factor
         end if
      else              !  Mail Proc
         dlrs=wgt
         write(40,31) rec, dlrs, bmcgrp, ldc, iw, actv
         ctmp1=ctmp1+1
         mp_bmc = mp_bmc + dlrs
         if ((actv.ne.6522).and.(bmcgrp.le.npool)) then
            ovh6522_bmc = ovh6522_bmc + (wgt*850133./849454.) !  Overhead factor
         end if
         if (bmcgrp.eq.75) then !  Breaks
            ct_brk_bmc = ct_brk_bmc + 1
            dlrs = wgt
```

```
                  brk_bmc = brk_bmc + dlrs
                  if (actv.ne.6522) then
                     ovh6522_bmc = ovh6522_bmc + (wgt*850133./849454.) !  Overhead factor
                  end if
               end if
            end if
         end if

:     Cost pool assignment for Non-MODS

         if (type.eq.'non') then
            nmodgrp = 0
            actv = if9806
            if (((if260.ge.0).and.(if260.le.8)).or.
     +         ((if260.ge.11).and.(if260.le.16)).or.
     +         ((if260.ge.18).and.(if260.le.23)).or.
     +         ((if260.ge.27).and.(if260.le.29))) then !  Mail Processing operation codes (F260)

               if (f9806.eq.'6521') then
                  nmodgrp = 75 !  Breaks
               else if (f128.eq.'A') then !  Manual
                  if (f9211.eq.'A') then
                     nmodgrp = 54 !  Manual Letters
                  else if (f9211.eq.'B') then
                     nmodgrp = 53 !  Manual Flats
                  else if (f9211.eq.'C') then
                     nmodgrp = 55 !  Manual Parcels
                  else
                     nmodgrp = 50 !  Allied Labor
                  end if
               else if ((f128.ge.'B').and.(f128.le.'F')) then
                  nmodgrp = 51 !  Automated distribution
               else if ((f128.ge.'G').and.(f128.le.'I')) then
                  nmodgrp = 50 !  Allied Labor
               else if ((f128.ge.'J').and.(f128.le.'M')) then
                  nmodgrp = 51 !  Automated distribution
               else if ((f128.ge.'N').and.(f128.le.'R')) then
                  nmodgrp = 50 !  Allied Labor
               else if (f128.eq.'S') then
                  nmodgrp = 51 ! Automated distribution
               else if ((f128.ge.'T').and.(f128.le.'U')) then
                  nmodgrp = 50 !  Allied Labor
               else if (((f116.ge.'A').and.(f116.le.'H')).or.
     &                ((f118.ge.'A').and.(f118.le.'K')).or.(f121.eq.'Y')) then
                  nmodgrp = 50 !  Allied Labor
               else if (if260.eq.18) then
                  nmodgrp = 56 !  Registry
               else if (if260.eq.22) then
                  nmodgrp = 52 !  Express
               else
                  nmodgrp = 57 !  Misc & support
               end if

c     Non-MODS encirclements

               if ((nmodgrp.eq.56).or.(nmodgrp.eq.57)) then !  Registry and Misc
                  actv = if9806
               else
                  actv = if9805
               end if

               if ((nmodgrp.eq.56).and.(actv.ne.60)) actv=if9805 ! Registry

               if (actv.eq.60) then
                  if ((f9805(2:4).eq.'510').and.((if9805.ge.1000).and.(if9805.le.4950))) then
                     actv = if9805
                  else if ((f9805(2:2).eq.'8').and.((if9805.ge.1000).and.(if9805.le.4950))) then
                     actv = if9805
                  else if ((f9805(1:3).ge.'545').and.(f9805(1:3).le.'548')) then
                     actv = if9805
                  else
                     actv = 60
                  end if
               end if
            end if

c  Reassign specific actv codes for expanded subclasses

            if ((actv.eq.1060).or.(actv.eq.2060).or.(actv.eq.3060).or.(actv.eq.4060)) then !  1st SP
               if (f136.eq.'D') then !  Metered
```

```
                write(50,31) rec, dlrs, nmodgrp, ldc, iw, actv !  Mail Proc tallies
                ctmp3=ctmp3+1
                mp_nmod = mp_nmod + dlrs
                if (nmodgrp.le.npool) then
                   if (actv.ne.6522) then
                      ovh6522_nmod = ovh6522_nmod + (wgt*4833550./4401822.) !  Overhead factor
                   end if
                   if ((actv.ne.6521).and.(actv.ne.6522)) then
                      ovhfact_nmod = ovhfact_nmod + (wgt*4833550./4401822.) !  Overhead factor
                   end if
                end if
             else
                print*, 'Pool not assigned f260 = ', if260
             end if
             if (nmodgrp.eq.75) then !  Z Breaks
                ct_brk_nmod = ct_brk_nmod + 1
                dlrs = wgt
                brk_nmod = brk_nmod + dlrs
                if (actv.ne.6522) then
                   ovh6522_nmod = ovh6522_nmod + (wgt*4833550./4401822.) !  Overhead factor
                end if
             end if
          else              !  Op code 88's now part of C/S 3.4
             write(56,21) rec !  Expedited delivery tallies
          end if
       end if
    end if

    end do

100 print*, 'Read exit error ', ier
    print*, 'Total Records ', ct
    print*, 'Number of obs used ', ctkeep
    print*, 'BMC  Total Obs ', ct1, ' Adm/Win ', ctaw1, ' MP ', ctmp1, ' Breaks ', ct_brk_bmc !
    print*, 'MODS Total Obs ', ct2, ' Adm/Win ', ctaw2, ' MP ', ctmp2 !
    print*, 'NMOD Total Obs ', ct3, ' Adm/Win ', ctaw3, ' MP ', ctmp3, ' Breaks ', ct_brk_nmod !
    print*, ' '
    print*, 'Number of MODS 1&2 tallies with a valid MODS code ', ct_good !
    print*, 'Number of MODS 1&2 tallies with an invalid MODS code ', ct_inv !
    print*, 'After residual pool assignment, number of tallies with invalid MODS codes ', ct_inva !
    print*, ' '
    print*, 'Total MODS 1&2 tally dollar weights ', cost_mod
    print*, 'Total BMCs tally dollar weights ', cost_bmc
    print*, 'Total Non-MODS tally dollar weights ', cost_nmod
    print*, ' '
    print*, 'Total MODS mail proc costs ', mp_mod
    print*, 'Total BMC mail proc costs ', mp_bmc
    print*, 'Total Non-MOD mail proc costs ', mp_nmod
    print*, ' '
    print*, 'Total MODS win tally costs = ', cost_win
    print*, 'Total MODS admin tally costs = ', cost_adm
    print*, 'Total MODS admin inq tally costs = ', cost_inq
    print*, 'Total MODS admin intl tally costs = ', cost_intl
    print*, 'Total MODS admin out tally costs = ', cost_out
    print*, 'Total BMCs admin/win tally costs = ', adm_bmc
    print*, 'Total BMC window costs = ', win_bmc
    print*, 'Total BMC break costs = ', brk_bmc
    print*, 'Total NMods admin/win tally costs = ', adm_non
    print*, 'Total NMods window costs = ', win_non
    print*, 'Total NMods break costs = ', brk_nmod
    print*, ' '
    print*, 'OVH6522 factor for BMCs (denominator) ', ovh6522_bmc
    print*, 'OVH6522 factor for Non-MODS (denominator) ', ovh6522_nmod
    print*, 'OVHFACT factor for Non-MODS (denominator) ', ovhfact_nmod
    print*, ' '
    print*, 'Registry checks '
    print*, 'Total Registry tallies before encirclement ', ct_reg_before
    print*, 'Total Registry tallies with F245, F246 = 006 ', ct_reg_006
    print*, 'Total Registry tallies after initial encirclement ', ct_reg_after
    print*, 'Total Registry tallies after LDC to F9805 encirclement ', ct_reg_ldc
    print*, 'Total Registry tallies after cost pool encirclement ', ct_reg_60
    print*, 'Total Registry tallies after F9606 encirclement ', ct_reg_f9606
    print*, 'Total Registry tallies after RPW encirclement ', ct_reg_rpw
    print*, 'Total Registry tallies after all encirclements ', ct_reg_final

    end

c   ------------------------------------------------------------------
c     pool   tcg 7/10/96
c     assigns pool groups to MODS numbers
```

```
      function   pool(mod)

      integer*4    pool
      character*3  mod

       pool = 100

       OCR OPERATIONS
      if ((mod.eq.'046').or.
     &    ((mod.ge.'830').and.(mod.le.'837')).or.
     &    ((mod.ge.'840').and.(mod.le.'847')).or.
     &    ((mod.ge.'850').and.(mod.le.'857')).or.
     &    ((mod.ge.'880').and.(mod.le.'887'))) then
         pool = 3                 !  OCR
       else if ((mod.ge.'301').and.(mod.le.'304')) then
         pool = 3                 !  Intl/OCR

c      BCS OPERATIONS
       else if ((mod.eq.'047').or.
     &       ((mod.ge.'241').and.(mod.le.'251')).or.
     &       ((mod.ge.'603').and.(mod.le.'604')).or.
     &       ((mod.ge.'860').and.(mod.le.'869')).or.
     &       ((mod.ge.'870').and.(mod.le.'879')).or.
     &       ((mod.ge.'914').and.(mod.le.'917')).or.
     &       ((mod.ge.'970').and.(mod.le.'979'))) then
         pool = 1                 !  BCS
       else if (((mod.ge.'311').and.(mod.le.'312')).or.
     &       ((mod.ge.'315').and.(mod.le.'316'))) then
         pool = 1                 !  BCS Intl
       else if (((mod.ge.'260').and.(mod.le.'267')).or.
     &       ((mod.ge.'270').and.(mod.le.'279')).or.
     &       ((mod.ge.'280').and.(mod.le.'287')).or.
     &       ((mod.ge.'290').and.(mod.le.'299')).or.
     &       ((mod.ge.'890').and.(mod.le.'899')).or.
     &       ((mod.ge.'908').and.(mod.le.'911')).or.
     &       ((mod.ge.'918').and.(mod.le.'919')).or.
     &       ((mod.ge.'925').and.(mod.le.'926'))) then
         pool = 2                 !  BCS/DBCS
       else if ((mod.eq.'309').or.
     &       ((mod.ge.'313').and.(mod.le.'314')).or.
     &       ((mod.ge.'317').and.(mod.le.'319')).or.
     &       ((mod.ge.'356').and.(mod.le.'357'))) then
         pool = 2                 !  BCS/DBCS Intl

c      LSM OPERATIONS
       else if (((mod.ge.'080').and.(mod.le.'089')).or.
     &       (mod.eq.'091').or.
     &       ((mod.ge.'093').and.(mod.le.'099'))) then
         pool=6                   ! LSM
       else if ((mod.eq.'090').or.(mod.eq.'092')) then
         pool=6                   ! LSM Intl

c      FSM OPERATIONS
       else if (((mod.ge.'140').and.(mod.le.'148')).or.
     &       (mod.eq.'191').or.
     &       ((mod.ge.'194').and.(mod.le.'197')).or.
     &       ((mod.ge.'331').and.(mod.le.'338')).or.
     &       ((mod.ge.'421').and.(mod.le.'428')).or.
     &       ((mod.ge.'960').and.(mod.le.'967'))) then
         pool=4                   !  FSM 881
       else if ((mod.eq.'192').or.(mod.eq.'193')) then
         pool=4                   ! FSM Intl
       else if (((mod.ge.'441').and.(mod.le.'448')).or.
     &       (mod.eq.'450').or.(mod.eq.'451').or.
     &       ((mod.ge.'461').and.(mod.le.'468'))) then
         pool=5                   !  FSM 1000
       else if (((mod.ge.'305').and.(mod.le.'308')).or.
     &       ((mod.ge.'452').and.(mod.le.'453'))) then
         pool = 5                 !  FSM 1000 Intl

c      Mechanized sort-sack outside
       else if ((mod.ge.'238').and.(mod.le.'239')) then
         pool=10                  !  1Sacks_m
       else if (mod.eq.'349') then
         pool=10                  ! 1Sacks_m Intl

c      MECHANIZED PARCEL SORTER
       else if (mod.eq.'105') then
         pool=7                   !  Mecparc
```

```
      else if ((mod.ge.'107').and.(mod.le.'108')) then
         pool=7                    !  Mecparc Intl

       SMALL PARCEL BUNDLE SORTER
      else if (((mod.ge.'134').and.(mod.le.'137')).or.
     &       ((mod.ge.'254').and.(mod.le.'257')).or.
     &       ((mod.ge.'434').and.(mod.le.'437'))) then
         pool=8                    !  SPBS Oth
       else if (((mod.ge.'052').and.(mod.le.'054')).or.
     &       ((mod.ge.'056').and.(mod.le.'058')).or.
     &       ((mod.ge.'346').and.(mod.le.'347'))) then
         pool = 8                  !  SPBS Oth Intl
       else if (((mod.ge.'138').and.(mod.le.'139')).or.
     &       ((mod.ge.'258').and.(mod.le.'259')).or.
     &       ((mod.ge.'438').and.(mod.le.'439'))) then
         pool=9                    !  SPBS Prio
       else if ((mod.eq.'104').or.(mod.eq.'106')) then
         pool = 9                  !  SPBS Prio Intl

c      MANUAL FLAT OPERATIONS
       else if ((mod.eq.'060').or.
     &       ((mod.ge.'069').and.(mod.le.'070')).or.
     &       ((mod.ge.'070').and.(mod.le.'075')).or.
     &       (mod.eq.'170').or.(mod.eq.'175').or.
     &       ((mod.ge.'178').and.(mod.le.'179'))) then
         pool=11                   !  MANF
       else if ((mod.ge.'062').and.(mod.le.'063')) then
         pool=11                   ! MANF Intl

c      MANUAL LETTERS OPERATIONS
       else if (((mod.ge.'029').and.(mod.le.'030')).or.
     &       ((mod.ge.'040').and.(mod.le.'045')).or.
     &       (mod.eq.'150').or.(mod.eq.'160').or.
     &       ((mod.ge.'168').and.(mod.le.'169'))) then
         pool=12                   !  MANL
       else if ((mod.ge.'032').and.(mod.le.'033')) then
         pool=12                   !  MANL Intl

c      MANUAL PARCEL OPERATIONS
       else if ((mod.eq.'100').or.
     &       (mod.eq.'130').or.(mod.eq.'200')) then
         pool = 13                 !  MANP
       else if (((mod.ge.'102').and.(mod.le.'103')).or.
     &       ((mod.ge.'202').and.(mod.le.'207'))) then
         pool = 13                 !  MANP Intl

c      MANUAL PRIORITY
       else if ((mod.eq.'050').or.(mod.eq.'055')) then
         pool=14                   !  Priority

c      LDC15
       else if (((mod.ge.'381').and.(mod.le.'386')).or.
     &       (mod.eq.'771').or.
     &       ((mod.ge.'774').and.(mod.le.'776')).or.
     &       (mod.eq.'779')) then
         pool=15                   !   LDC 15

c      ALLIED OPERATIONS

c      ACDCS
       else if ((mod.eq.'064').or.
     &       ((mod.ge.'118').and.(mod.le.'119'))) then
         pool=23                   !  1Scan
       else if (mod.eq.'350') then
         pool = 23                 !  1Scan Intl

c      Bulk presort
       else if ((mod.ge.'002').and.(mod.le.'009')) then
         pool=16                   !  1Bulk Pr

c      Cancellation/mail prep
       else if ((mod.ge.'010').and.(mod.le.'028')) then
         pool=17                   !  1CancMPP

c      opening unit - pref
       else if (((mod.ge.'110').and.(mod.le.'114')).or.
     &       ((mod.ge.'180').and.(mod.le.'184'))) then
         pool=19                   !  1OpPref
       else if ((mod.ge.'343').and.(mod.le.'344')) then
         pool = 19                 !  1OpPref Intl
```

```
      else if ((mod.ge.'358').and.(mod.le.'359')) then
         pool = 19                 !  1OpPref (1Robotic)

:     opening unit - bbm
      else if (((mod.ge.'115').and.(mod.le.'117')).or.
     &     ((mod.ge.'185').and.(mod.le.'189'))) then
         pool=18                   !  1OpBulk

:     pouching
      else if (((mod.ge.'120').and.(mod.le.'129')).or.
     &     ((mod.ge.'208').and.(mod.le.'209'))) then
         pool=21                   !  1Pouching
      else if (mod.eq.'345') then
         pool = 21                 !  1Pouching Intl

:     platform
      else if ((mod.ge.'210').and.(mod.le.'234')) then
         pool=20                   !  1Platform
      else if (((mod.ge.'351').and.(mod.le.'352')).or.
     &     (mod.eq.'454')) then
         pool = 20                 ! 1Platform Intl

c     manual sack sort
      else if ((mod.ge.'235').and.(mod.le.'237')) then
         pool=22                   !  1Sacks_h
      else if (mod.eq.'348') then
         pool = 22                 !  1Sacks_h Intl

c     DAMAGED PARCEL REWRAP
      else if (mod.eq.'109') then
         pool=28                   !  Rewrap
      else if (mod.eq.'574') then
         pool = 28                 !  Rewrap Intl

c     EXPRESS
      else if ((mod.eq.'131').or.(mod.eq.'669').or.(mod.eq.'793')) then
         pool=25                   !  Express
      else if (mod.eq.'575') then
         pool = 25                 !  Express Intl

c     empty equipment
      else if (mod.eq.'549') then
         pool=29                   !  1EEqmt
      else if (mod.eq.'576') then
         pool = 29                 ! 1EEqmt Intl

c     MAILGRAM
      else if (mod.eq.'584') then
         pool=26                   !  Mailgram

c     MAIL PROCESSING SUPPORT
      else if (((mod.ge.'340').and.(mod.le.'341')).or.
     &     (mod.eq.'547').or.(mod.eq.'548').or.
     &     ((mod.ge.'554').and.(mod.le.'555')).or.
     &     (mod.eq.'607').or.
     &     (mod.eq.'612').or.(mod.eq.'620').or.(mod.eq.'630').or.
     &     (mod.eq.'677').or.(mod.eq.'755').or.(mod.eq.'798')) then
         pool=31                   !  1Support

c     MISCELLANEOUS
      else if ((mod.ge.'560').and.(mod.le.'564')) then
         pool=30                   !  1Misc
      else if ((mod.eq.'132').or.
     &     ((mod.ge.'545').and.(mod.le.'546')).or.
     &     (mod.eq.'577').or.(mod.eq.'580').or.(mod.eq.'681')) then
         pool = 30                 !  1Misc Intl

c     BUSINESS REPLY / POSTAGE DUE
      else if (mod.eq.'930') then
         pool=24                   !  Bus Reply
      else if (mod.eq.'573') then
         pool = 24                 !  Bus Reply Intl

c     REGISTRY
      else if ((mod.ge.'585').and.(mod.le.'590')) then
         pool=27                   !  Registry
      else if (mod.eq.'578') then
         pool = 27                 !  Registry Intl

c     LDC41 AND LDC42
```

```
      else if ((mod.eq.'048').or.(mod.eq.'049').or.
     &       (mod.eq.'252').or.(mod.eq.'253').or.
     &       ((mod.ge.'361').and.(mod.le.'362')).or.
     &       ((mod.ge.'364').and.(mod.le.'366')).or.
     &       ((mod.ge.'371').and.(mod.le.'378')).or.
             ((mod.ge.'411').and.(mod.le.'417')).or.
             ((mod.ge.'605').and.(mod.le.'606')).or.
     &       ((mod.ge.'821').and.(mod.le.'829')).or.
     &       ((mod.ge.'905').and.(mod.le.'907')).or.
     &       ((mod.ge.'912').and.(mod.le.'913')).or.
     &       ((mod.ge.'942').and.(mod.le.'943'))) then
          pool=33                  ! LDC 41
       else if (((mod.ge.'400').and.(mod.le.'407')).or.
     &       ((mod.ge.'801').and.(mod.le.'819'))) then
          pool=34                  !  LDC 42

c       MANUAL DISTRIBUTION - STATION/BRANCH (LDC43)
       else if (mod.eq.'240') then
          pool=35                  ! LDC 43

c       STATION/BRANCH - BOX SECTION (LDC44)
       else if (mod.eq.'769') then
          pool=36                  !  LDC 44

c       WINDOW Service
       else if ((mod.eq.'355').or.(mod.eq.'568')) then
          pool=95

c       LDC48
       else if (mod.eq.'583') then
          pool=37                  !  LDC48 Exp
       else if ((mod.eq.'353').or.(mod.eq.'558').or.(mod.eq.'559').or.
     &       (mod.eq.'608').or.(mod.eq.'621').or.
     &       (mod.eq.'631').or.(mod.eq.'678')) then
          pool=39                  !  LDC48 Adm
       else if ((mod.ge.'542').and.(mod.le.'544')) then
          pool=40                  !  LDC48 SSV
       else if ((mod.eq.'741').or.(mod.eq.'742').or.(mod.eq.'794')) then
          pool=38                  !  LDC48 Oth

c       ADDRESS INFO SYSTEM & CENTRAL MAIL MARK-UP
       else if ((mod.eq.'539').or.
     &       ((mod.ge.'791').and.(mod.le.'792')).or.
     &       ((mod.ge.'795').and.(mod.le.'797'))) then
          pool=41                  !  LDC 49

c       MAILING REQUIREMENTS & BUSINESS MAIL ENTRY
       else if ((mod.eq.'001').or.(mod.eq.'550').or.(mod.eq.'660').or.
     &       (mod.eq.'697')) then
          pool=42                  !  LDC 79

c       invalid mods code for mail processing
       else
          pool = 100
       end if

       if (pool.eq.100) then

c       ADMINISTRATION

c       2adm_out
          if ((mod.eq.'342').or.(mod.eq.'354').or.((mod.ge.'455').and.(mod.le.'459')).or.
     &       ((mod.ge.'471').and.(mod.le.'504')).or.((mod.ge.'599').and.(mod.le.'602')).or.
     &       ((mod.ge.'613').and.(mod.le.'614')).or.(mod.eq.'616').or.(mod.eq.'622').or.
     &       (mod.eq.'624').or.(mod.eq.'632').or.((mod.ge.'634').and.(mod.le.'635')).or.
     &       (mod.eq.'641').or.(mod.eq.'655').or.(mod.eq.'671').or.(mod.eq.'676').or.
     &       ((mod.ge.'698').and.(mod.le.'703')).or.((mod.ge.'609').and.(mod.le.'703')).or.
     &       ((mod.ge.'705').and.(mod.le.'740')).or.((mod.ge.'743').and.(mod.le.'754')).or.
     &       ((mod.ge.'757').and.(mod.le.'762')).or.(mod.eq.'768').or.(mod.eq.'770').or.
     &       ((mod.ge.'920').and.(mod.le.'924')).or.((mod.ge.'927').and.(mod.le.'929')).or.
     &       ((mod.ge.'932').and.(mod.le.'937')).or.((mod.ge.'946').and.(mod.le.'953'))) then
             pool = 99
          end if

c       2Adm
          if (((mod.ge.'505').and.(mod.le.'538')).or.((mod.ge.'540').and.(mod.le.'541')).or.
     &       ((mod.ge.'556').and.(mod.le.'557')).or.(mod.eq.'566').or.
     &       ((mod.ge.'569').and.(mod.le.'572')).or.(mod.eq.'579').or.
     &       ((mod.ge.'581').and.(mod.le.'582')).or.((mod.ge.'591').and.(mod.le.'596')).or.
     &       ((mod.ge.'610').and.(mod.le.'611')).or.(mod.eq.'615').or.(mod.eq.'617').or.
```

```
&       (mod.eq.'623').or.(mod.eq.'633').or.(mod.eq.'636').or.
&       ((mod.ge.'642').and.(mod.le.'643')).or.((mod.ge.'645').and.(mod.le.'654')).or.
&       ((mod.ge.'656').and.(mod.le.'659')).or.((mod.ge.'661').and.(mod.le.'666')).or.
&       (mod.eq.'668').or.(mod.eq.'670').or.((mod.ge.'672').and.(mod.le.'675')).or.
        ((mod.ge.'679').and.(mod.le.'680')).or.((mod.ge.'682').and.(mod.le.'687')).or.
        (mod.eq.'689').or.((mod.ge.'691').and.(mod.le.'696')).or.(mod.eq.'704').or.
        ((mod.ge.'763').and.(mod.le.'766')).or.((mod.ge.'772').and.(mod.le.'773')).or.
&       (mod.eq.'704').or.((mod.ge.'780').and.(mod.le.'789')).or.
&       ((mod.ge.'900').and.(mod.le.'904')).or.((mod.ge.'958').and.(mod.le.'959')).or.
&       ((mod.ge.'968').and.(mod.le.'969')).or.((mod.ge.'980').and.(mod.le.'987'))) then
        pool = 98
    end if

  2Adm inq (Claims and Inquiry)
    if ((mod.ge.'551').and.(mod.le.'552')) then
       pool = 97
    end if

 end if

 return
 end
```

Section II: POSTAL SERVICE Method Volume-Variable Cost Estimates by Weight Increment– Clerks and Mailhandlers, Mail Processing

(Programs: modsproc00_wgt.f, sumclass_mod_wgt.f, bmcproc00_wgt.f, sumclass_bmc_wgt.f, nmodproc00_wgt.f, sumclass_nmod_wgt.f)

```
program modsproc00_wgt

Purpose:  Computes distributed volume-variable costs (USPS Method) for MODS 1&2 offices
          Adds additional dimension for various weight categories

 nplicit none

integer*4 nmod, nw, nmod2, nw2
integer*4 nact, nshp, nmix, nmixcl, nact2
integer*4 nitem, nshp2, ncsi, ncon, begmail

parameter (nmod = 43)      !  Number of cost pools (includes the LDC 15 Proxy cost pool)
parameter (nmod2 = 42)     !  Number of distribution cost pools
parameter (nw =  22)       !  Number of weight increments (including no weight)
parameter (nw2 = 21)       !  Number of weight increments
parameter (nact = 255)     !  Number of direct activity codes
parameter (nshp = 6)       !  Number of shapes
parameter (nitem = 16)     !  Number of item types
parameter (nshp2 = 5)      !  Number of shapes (not including other)
parameter (ncon = 10)      !  Number of container types
parameter (nmix = 20)      !  Number of combined activity codes - for dist of counted items
parameter (ncsi = nshp2 + nitem) !  Number of "identified" container types (loose shapes + items)
parameter (begmail = 17)   ! Set this to the index of the first non-Spec Serv activity code
parameter (nmixcl = 20)    ! Number of class-specific mixed-mail codes
parameter (nact2 = 275)    ! Number of activity codes including class-specific mixed-mail

include 'iocs2000.h'

real*8     adols(nw,nmod,nact2,nshp) ! Handling direct single piece
real*8    adist(nw,nmod,nact2,nshp) ! Workspace for distribution of no weight single pieces
real*8     bdols(nw,nmod,nitem,nact2) ! Handling identical or top-piece item
real*8    bdist(nw,nmod,nitem,nact2) ! Workspace for distribution of no weight identical/top-piece items
real*8    cdist(nw,nmod,nitem,nact2) ! Workspace for distribution of matrix D
real*8     cdols(nw,nmod,nitem,nact2) ! Workspace for distributed costs from matrix D
real*8     ddols(nmod,nitem) ! Handling mixed/empty item
real*8     fdols(nw,nmod,ncon,nact2) ! Handling identical or top-piece container
real*8    fdist(nw,nmod,ncon,nact2) ! Workspace for distribution of no weight identical/top-piece containers
real*8     gdols(nmod,ncon,ncsi) ! Handling "identified" container
real*8    gdist(nw,nmod,ncon,nact2) ! G Matrix distributed to activity code
real*8     hdols(nmod,ncon) ! Handling uncounted/empty container
real*8    result(nw,nmod,nact2) ! Array to hold results
real*8    resulta(nw,nmod,nact2) ! Array to hold results for matrix A
real*8    resultb(nw,nmod,nact2) ! Array to hold results for matrix B, C, D
real*8    resultf(nw,nmod,nact2) ! Array to hold results for matrix F, G, H
real*8    resultj(nw,nmod,nact2) ! Array to hold distributed J matrix
real*8    work(nw,nmod,nact2) ! Array to hold distributed mixed class-specific
real*8    jdols(nmod)      ! Not Handling
real*8    counts(ncsi)
real*8    actshr(nw,nact2), actshr3(nact), actwgt(nw2), actshr2(nw,nact2)
real*8    dlrs, sum, distsum, rf9250, tot_dol, tot_dol2, check, totj
real*8    atot, btot, ctot, dtot, ftot, gtot, htot, jtot
real*8    pooldols(nmod)
real*8    variable(nmod), wincost(nact,nw)
real*8    varcost(nw,nmod,nact)
real*8    novarcst(nw,nmod,nact)
real*8    distsum48, sum48, cost_cntr, cost_unid

logical flag

integer*4 acnt, bcnt, ccnt, dcnt, fcnt, gcnt, hcnt, jcnt
integer*4 ind, if114, ldc, k, l
integer*4 cnt, class(nact2), class_code(nact2)
integer*4 i, j, imat, imod, icon, iact, icsi, iitem, shapeind, iw
integer*4 ier, ct_cntr, ct_unid, ishp
integer*4 mapcodes(20)
integer*4 searchc, searchi, modgrp, hand, actv
integer*4 mixcodes(nmixcl)
integer*4 acodes(nact2), mixcount(nmixcl)
integer*4 mixmap(nact,nmixcl)
integer*4 ldc1(nmod)
integer*4 if166, if167, weight, ct_nowgt

 haracter*14 modcodes(nmod)
 haracter*1 codes(26)/'A','B','C','D','E','F','G','H','I','J','K',
&   'L','M','N','O','P','Q','R','S','T','U','V',
&   'W','X','Y','Z'/

logical flag2

atot = 0.0
```

```
      btot = 0.0
      ctot = 0.0
      dtot = 0.0
      ftot = 0.0
      gtot = 0.0
      htot = 0.0
      jtot = 0.0
      acnt = 0
      bcnt = 0
      ccnt = 0
      dcnt = 0
      fcnt = 0
      gcnt = 0
      hcnt = 0
      jcnt = 0
      cnt = 0
      ier = 0

      do i = 1, nmod
         pooldols(i) = 0.0
         variable(i) = 0.0
      end do

      do i = 1, 20
         mapcodes(i) = 0
      end do

      do i = 1, nmixcl
         mixcodes(i) = 0
         mixcount(i) = 0
      end do

c     Map of activity codes
      open(20,file='activity00.ecr.cra')
21    format(i4,i6,i5)
      do i=1,nact2
         read (20,21) acodes(i), class(i), class_code(i)
      end do
      print *,'read activity map'
      close(20)

c     Map of class specific mixed-mail activity codes
      open(20,file='mixclass.intl')
      do i = 1,nmixcl
         read (20,21) mixcodes(i)
      end do
      print *,'read mixed item code list'
      close(20)

      do i = 1,nact
         do j = 1,nmixcl
            mixmap(i,j) = 0
         end do
      end do

c     Maps class specific mixed-mail activity codes to appropriate direct activity codes
      open(20,file='mxmail.intl.dat')
23    format(20i4)

      do while (ier.eq.0)
         read (20,23,iostat=ier,end=75) mapcodes
         i = searchi(mixcodes,nmixcl,mapcodes(1))
         if (i.gt.0) then
            flag = .true.
            ind = 1
            do while ((flag).and.(ind.lt.20))
               ind = ind + 1
               if (mapcodes(ind).gt.0) then
                  j = searchi(acodes,nact,mapcodes(ind))
                  if (j.gt.0) then
                     mixcount(i) = mixcount(i) + 1
                     mixmap(mixcount(i),i) = j
                  else
                     print *,' Direct mail code did not map ',mapcodes(ind)
                  end if
               else
                  flag = .false.
               end if
            end do
         else
```

```
         print *,' Mixed mail code did not map ',mapcodes(1)
      end if
   end do
   print *,' read mixed-mail map with exit code = ',ier

   close(20)

   Map of cost pool dollars and variabilities by cost pool
   open(20,file='costpools.00.619')
   format(2x,a16,i2,f10.0,f6.2)
   do i = 1,nmod
      read(20,24) modcodes(i), ldc1(i), pooldols(i), variable(i)
   end do

   close(20)

   Window Service costs by activity code and weight category used in Funtion 4 support cost distribution
   open(20,file='windk_wgt_ecr.00.619')
   format(7x,f16.5)

   do i = 1, nact
      do iw = 1, nw
         read(20,28) wincost(i,iw)
      end do
   end do
   print*, 'MODS Window Service costs read in '
   close(20)

Initialize matrices

   do iw = 1,nw
      do imod = 1,nmod
         do iact = 1,nact
            varcost(iw,imod,iact) = 0.
            novarcst(iw,imod,iact) = 0.
         end do
      end do
   end do
   do ishp = 1, nshp
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               adols(iw,imod,iact,ishp) = 0.0
               adist(iw,imod,iact,ishp) = 0.0
            end do
         end do
      end do
   end do
   do iact = 1, nact2
      do iitem = 1, nitem
         do imod = 1, nmod
            do iw = 1, nw
               bdols(iw,imod,iitem,iact) = 0.
               bdist(iw,imod,iitem,iact) = 0.
            end do
         end do
      end do
   end do
   do iact = 1, nact2
      do iitem = 1, nitem
         do imod = 1, nmod
            do iw = 1, nw
               cdist(iw,imod,iitem,iact) = 0.
            end do
         end do
      end do
   end do
   do iact = 1, nact2
      do iitem = 1, nitem
         do imod = 1, nmod
            do iw=1,nw
               cdols(iw,imod,iitem,iact) = 0.
            end do
         end do
      end do
   end do
   do iitem = 1, nitem
      do imod = 1, nmod
         ddols(imod,iitem) = 0.
      end do
```

```
      end do
      do iact = 1, nact2
         do icon = 1, ncon
            do imod = 1, nmod
               do iw = 1, nw
                  fdols(iw,imod,icon,iact) = 0.
                  fdist(iw,imod,icon,iact) = 0.
               end do
            end do
         end do
      end do
      do icsi = 1, ncsi
         do icon = 1, ncon
            do imod = 1, nmod
               gdols(imod,icon,icsi) = 0.
            end do
         end do
      end do
      do iact = 1, nact2
         do icon = 1, ncon
            do imod = 1, nmod
               do iw = 1, nw
                  gdist(iw,imod,icon,iact) = 0.
               end do
            end do
         end do
      end do
      do icon = 1, ncon
         do imod = 1, nmod
            hdols(imod,icon) = 0.
         end do
      end do
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               result(iw,imod,iact) = 0.
            end do
         end do
      end do
      o iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               resulta(iw,imod,iact) = 0.
            end do
         end do
      end do
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               resultb(iw,imod,iact) = 0.
            end do
         end do
      end do
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               resultf(iw,imod,iact) = 0.
            end do
         end do
      end do
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               work(iw,imod,iact) = 0.
               resultj(iw,imod,iact) = 0.
            end do
         end do
      end do
      do imod = 1, nmod
         jdols(imod) = 0.
      end do
      print*, 'Matrices initialized '

      pen(25,file='mods12_mp00by_new.dat',recl=1200) !  MODS 1&2 offices mail proc IOCS data
31    ormat(a167,f15.5,i2,i2,i3,i5)

      cnt = 0
      ier = 0
      tot_dol = 0.0
      tot_dol2 = 0.0
```

```
      totj =  0.0
      ct_cntr = 0
      ct_unid = 0
      cost_cntr = 0.0
      cost_unid = 0.0
        _nowgt = 0

      do while (ier.eq.0)

         read(25,31,iostat=ier,end=100) rec,dlrs,modgrp,ldc,iw,actv

         cnt = cnt + 1
         iw = 1

         read(f114,'(i3)') if114
         read(f9250,'(f10.0)') rf9250
         read(f166,'(i2)') if166
         read(f167,'(i2)') if167

         dlrs = rf9250/100000.

         tot_dol = tot_dol + dlrs

         ishp = shapeind(actv,f9635,f9805) !  Subroutine assigns shape

c     Break out Std A ECR Saturation and High Density into separate activity codes
         if ((actv.eq.1311).or.(actv.eq.2311).or.(actv.eq.3311).or.(actv.eq.4311)) then !  Std A  WSH/WSS
            if (f9619.eq.'1') then !  WSS
               if (actv.eq.1311) actv = 1313
               if (actv.eq.2311) actv = 2313
               if (actv.eq.3311) actv = 3313
               if (actv.eq.4311) actv = 4313
            end if
         end if

         if ((actv.eq.1331).or.(actv.eq.2331).or.(actv.eq.3331).or.(actv.eq.4331)) then !  Std A NP WSH/WSS
            if (f9619.eq.'1') then !  WSS
               if (actv.eq.1331) actv = 1333
               if (actv.eq.2331) actv = 2333
               if (actv.eq.3331) actv = 3333
               if (actv.eq.4331) actv = 4333
            end if
         end if

c     Any "auto" ECR flats or parcels are assumed to be basic ECR
         if (actv.eq.2312) actv = 2310
         if (actv.eq.3312) actv = 3310
         if (actv.eq.4312) actv = 4310
         if (actv.eq.2332) actv = 2330
         if (actv.eq.3332) actv = 3330
         if (actv.eq.4332) actv = 4330

c     Assign handling category
         if (((actv.ge.1000).and.(actv.le.4950)).or.((actv.ge.5300).and.(actv.le.5480))) then
            hand = 1             ! direct (non special services)
         else if ((actv.ge.10).and.(actv.lt.1000)) then
            if (((f9805.ge.'1000').and.(f9805.le.'4950')).or.
     &          ((f9805(1:2).ge.'53').and.(f9805(1:2).le.'54'))) then
               hand = 1          ! direct (non special service handling)
            else if ((f9635.ge.'A').and.(f9635.le.'K')) then
               hand = 1          ! direct (special services)
            else if ((f9214.ge.'A').and.(f9214.le.'P')) then
               hand = 2          ! mixed item
            else if ((f9219.ge.'A').and.(f9219.le.'J')) then
               hand = 3          ! mixed container
            else
               hand = 4          !  not handling mail
            end if
         else if ((f9214.ge.'A').and.(f9214.le.'P')) then
            hand = 2             ! mixed item
         else if ((f9219.ge.'A').and.(f9219.le.'J')) then
            hand = 3             ! mixed container
         else
            hand = 4             ! not handling mail
         end if

         iitem = searchc(codes,nitem,f9214) !  Assign item type
         icon = searchc(codes,ncon,f9219) !  Assign container type
         iact = searchi(acodes,nact2,actv) !  Activity codes
```

```
      Assign weight increment
         if (hand.eq.1) then
            if (actv.ge.1000) then
               iw = weight(f165,if166,if167,ct_nowgt,nw) !  Subroutine assigns weight increment
            else
               iw = nw           !  Special service activities assumed to have no record weight
            end if
         else
            iw = nw
         end if

         if ((hand.eq.1).and.(((if114.ge.271).and.(if114.le.278)).or.
     &       ((if114.ge.971).and.(if114.le.978)))) then
            result(iw,nmod,iact) = result(iw,nmod,iact) + dlrs !  LDC 15 Proxy distrib key using BCS, DBCS MODS codes
         end if

         if ((hand.eq.1).and.(iact.eq.0)) then
            print *,'missing direct activity code = ',actv,' modgrp = ',modgrp
         end if

C     Single piece being handled, Assign to A matrix
         if ((hand.eq.1).and.(iitem.eq.0).and.(icon.eq.0)) then
            if (iact.gt.0) then
               if ((modgrp.gt.0).and.(modgrp.le.nmod)) then
                  adols(iw,modgrp,iact,ishp)=adols(iw,modgrp,iact,ishp) + dlrs !  Direct single piece
                  atot = atot + dlrs
                  acnt = acnt + 1
                  tot_dol2 = tot_dol2 + dlrs
               else
                  print *,' bad MODS in matrix A ',f114, modgrp, dlrs
               end if
            else                 !  Not handling mail
               print *,'Not-handling tally with direct code = ',actv,' cost pool = ',modgrp
               if ((modgrp.gt.0).and.(modgrp.ne.15)) then !  Exclude LDC 15
                  jdols(modgrp) = jdols(modgrp) + dlrs
                  jtot = jtot + dlrs
                  jcnt = jcnt + 1
                  tot_dol2 = tot_dol2 + dlrs
               end if
            end if

C*****************************************************************************
C     Not-handling mail tallies -- assign to J matrix
         else if (hand.eq.4) then
            if (modgrp.ne.15) then !  Exclude LDC 15
               jdols(modgrp) = jdols(modgrp) + dlrs
               jtot = jtot + dlrs
               jcnt = jcnt + 1
               tot_dol2 = tot_dol2 + dlrs
            else
               totj = totj + dlrs
            end if

C*****************************************************************************
C     Item being handled: separate items with direct activity codes from others

         else if ((f9214.ge.'A').and.(f9214.le.'P')) then
            if (hand.eq.1) then
               imat = 1          ! "B" matrix - identical, top piece, or counted item
            else if (hand.eq.2) then
               imat = 3          ! "D" matrix - mixed, empty item
            else
               print *,'problem item in modgrp = ',modgrp
               imat = 0
            end if

C     "D" matrix: mixed or empty item
            if (imat.eq.3) then
               ddols(modgrp,iitem) = ddols(modgrp,iitem) + dlrs
               dtot = dtot + dlrs
               dcnt = dcnt + 1
               tot_dol2 = tot_dol2 + dlrs

C     "B" matrix: identical or top piece rule (direct item)
            else if (imat.eq.1) then
               bdols(iw,modgrp,iitem,iact) =
     &            bdols(iw,modgrp,iitem,iact) + dlrs
               btot = btot + dlrs
               bcnt = bcnt + 1
               tot_dol2 = tot_dol2 + dlrs
```

```
          else                  !  Not handling mail
             print *,' imat 0 in modgrp = ',actv
             jdols(modgrp) = jdols(modgrp) + dlrs
             jtot = jtot + dlrs
             jcnt = jcnt + 1
             tot_dol2 = tot_dol2 + dlrs
          end if

C*****************************************End Item***************************
C     Container being handled: separate containers with direct activity codes from others
         else if (icon.gt.0) then

            if (modgrp.gt.0) then

               ct_cntr = ct_cntr + 1
               cost_cntr = cost_cntr + dlrs

               flag2=.false.

               if (f9901(1:1).eq.'%') then
                  read(rec(340:406),451,iostat=ier) counts
               else
                  read(rec(339:406),450,iostat=ier) counts
               end if
450            format(5(1x,f3.0),16f3.0)
451            format(f3.0,4(1x,f3.0),16f3.0)


               if (ier.ne.0) then
                  flag2 = .true.
                  j = 340
                  do i = 1, ncsi
                     counts(i) = 0.
                  end do
                  ier = 0
               end if

               sum = 0.
               do i = 1, ncsi
                  sum = sum + counts(i)
               end do

c     "F" matrix: identical mail in container (direct container)
               if (hand.eq.1) then
                  fdols(iw,modgrp,icon,iact) =
     &               fdols(iw,modgrp,icon,iact) + dlrs
                  ftot = ftot + dlrs
                  fcnt = fcnt + 1
                  tot_dol2 = tot_dol2 + dlrs

c     "H" matrix:  Uncounted, empty, or contents read error
               else if ((sum.eq.0.).or.flag2) then
                  hdols(modgrp,icon) = hdols(modgrp,icon) + dlrs
                  htot = htot + dlrs
                  hcnt = hcnt + 1
                  tot_dol2 = tot_dol2 + dlrs
                  if (actv.ne.6523) then
                     ct_unid = ct_unid + 1
                     cost_unid = cost_unid + dlrs
                  end if

c     "G" matrix:  container contents are "identified"
               else if (sum.gt.0.) then
                  do icsi = 1, ncsi
                     gdols(modgrp,icon,icsi) = gdols(modgrp,icon,icsi) +
     &                  (counts(icsi)/sum) * dlrs
                  end do
                  gtot = gtot + dlrs
                  gcnt = gcnt + 1
                  tot_dol2 = tot_dol2 + dlrs
               end if
            else
               print *,' bad container or mods code ',f9219,',',modgrp
               if ((modgrp.gt.0).and.(modgrp.ne.15)) then !  LDC 15
                  jdols(modgrp) = jdols(modgrp) + dlrs
                  jtot = jtot + dlrs
                  jcnt = jcnt + 1
                  tot_dol2 = tot_dol2 + dlrs
               end if
            end if
```

```
!************************************End Container*********************************
:     Any remaining tallies considered not handling mail
         else
            print *,'Shouldnt get here resid J'
            if (modgrp.ne.15) then !  LDC 15
               jdols(modgrp) = jdols(modgrp) + dlrs
               jtot = jtot + dlrs
               jcnt = jcnt + 1
               tot_dol2 = tot_dol2 + dlrs
            end if
         end if

      end do
100   print *,' read exit = ',ier,' with ',cnt,' records ', ' dlrs = ', tot_dol
      print*, 'Total assigned dlrs = ', tot_dol2, ' j dols for LD 15 ', totj

C ************************************End Read Loop*********************************

c     Redistribute no weight direct single piece costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do ishp = 1, nshp
               if (adols(nw,imod,iact,ishp).gt.0.0) then
                  sum = 0.0
                  do iw = 1, nw2 !  Distribute over all weight increments
                     sum = sum + adols(iw,imod,iact,ishp)
                  end do
                  if (sum.gt.0) then
                     do iw = 1, nw2
                        adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                     adols(nw,imod,iact,ishp)*adols(iw,imod,iact,ishp)/sum
                     end do
                     adols(nw,imod,iact,ishp) = 0.0
                  end if
               end if
            end do
         end do
      end do

c     Residual distribution of direct single piece no weight costs

      do iact = begmail, nact2
         do imod = 1, nmod
            do ishp = 1, nshp
               if (adols(nw,imod,iact,ishp).gt.0.0) then
                  sum = 0.0
                  do iw = 1, nw2
                     actwgt(iw) = 0.0
                  end do
                  do j = 1, nmod2 !  Distbribute over all cost pools (exclude LDC 15 proxy (pool #43))
                     do iw = 1, nw2 !  Distribute over all weight increments
                        actwgt(iw) = actwgt(iw) + adols(iw,j,iact,ishp)
                        sum = sum + adols(iw,j,iact,ishp)
                     end do
                  end do
                  if (sum.gt.0) then
                     do iw = 1, nw2
                        adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                     adols(nw,imod,iact,ishp)*actwgt(iw)/sum
                     end do
                     adols(nw,imod,iact,ishp) = 0.0
                  else
                     if (adols(nw,imod,iact,ishp).gt.0.) then
                        print*,'Level 3a distribution of act = ',acodes(iact)
                        do k = begmail, nact2
                           do iw = 1, nw2
                              actshr2(iw,k) = 0.
                           end do
                        end do
                        do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                           if (class(k).eq.class(iact)) then !  Same subclass
                              do iw = 1, nw2 !  Distribute over all weight increments
                                 actshr2(iw,k) = actshr2(iw,k) + adols(iw,imod,k,ishp)
                                 sum = sum + adols(iw,imod,k,ishp)
                              end do
                           end if
                        end do
                        if (sum.gt.0.) then
                           do k = begmail, nact2
```

```
                        do iw = 1, nw2
                           adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                        adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                        end do
                     end do
                     adols(nw,imod,iact,ishp) = 0.0
                  else
                     print*,'Level 4a distribution of act = ',acodes(iact)
                     do k = begmail, nact2
                        do iw = 1, nw2
                           actshr2(iw,k) = 0.
                        end do
                     end do
                     do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                        if (class(k).eq.class(iact)) then !  Same subclass
                           do j = 1,nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                              do iw = 1, nw2 !  Disbribute over all weight increments
                                 actshr2(iw,k) = actshr2(iw,k) + adols(iw,j,k,ishp)
                                 sum = sum + adols(iw,j,k,ishp)
                              end do
                           end do
                        end if
                     end do
                     if (sum.gt.0.) then
                        do k = begmail, nact2
                           do iw = 1, nw2
                              adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                           adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                           end do
                        end do
                        adols(nw,imod,iact,ishp) = 0.0
                     else
                        print*,'unable to distribute no weight for ',
     &                     imod,' act = ',acodes(iact), ' cost = ', adols(nw,imod,iact,ishp)
                     end if
                  end if
               end if
            end if
         end if
      end do
   end do
end do

c     Add in redistributed no weight direct single piece costs
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               do ishp = 1, nshp
                  adols(iw,imod,iact,ishp) = adols(iw,imod,iact,ishp) + adist(iw,imod,iact,ishp)
               end do
            end do
         end do
      end do

c     Redistribute no weight identical/top piece item costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do iitem = 1, nitem
               if (bdols(nw,imod,iitem,iact).gt.0) then
                  sum = 0.0
                  do iw = 1, nw2 !  Distribute over all weight increments
                     sum = sum + bdols(iw,imod,iitem,iact)
                  end do
                  if (sum.gt.0.0) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                     bdols(nw,imod,iitem,iact)*bdols(iw,imod,iitem,iact)/sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  end if
               end if
            end do
         end do
      end do

c     Residual distribution of identical/top piece items no weight costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do iitem = 1, nitem
               if (bdols(nw,imod,iitem,iact).gt.0.0) then
```

```
                sum = 0.0
                do iw = 1, nw2
                   actwgt(iw) = 0.0
                end do
                do j = 1, nitem !  Distribute over all item types
                   do iw = 1, nw2 !  Distribute over all weight increments
                      actwgt(iw) = actwgt(iw) + bdols(iw,imod,j,iact)
                      sum = sum + bdols(iw,imod,j,iact)
                   end do
                end do
                if (sum.gt.0) then
                   do iw = 1, nw2
                      bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                   bdols(nw,imod,iitem,iact)*actwgt(iw)/sum
                   end do
                   bdols(nw,imod,iitem,iact) = 0.0
                else
                   if (bdols(nw,imod,iitem,iact).gt.0.0) then
                      print*, 'Level 3 b distribution of act = ', acodes(iact)
                      do iw = 1, nw2
                         actwgt(iw) = 0.
                      end do
                      do k = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                         do j = 1,nitem !  Distribute over all item types
                            do iw = 1, nw2 !  Distribute over all weight increments
                               actwgt(iw) = actwgt(iw) + bdols(iw,k,j,iact)
                               sum = sum + bdols(iw,k,j,iact)
                            end do
                         end do
                      end do
                      if (sum.gt.0.) then
                         do iw = 1, nw2
                            bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                         bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                         end do
                         bdols(nw,imod,iitem,iact) = 0.0
                      else
                         print*, 'Level 4 b distribution of act = ', acodes(iact)
                         do iw = 1, nw2
                            actwgt(iw) = 0.
                         end do
                         do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                            if (class(k).eq.class(iact)) then !  Same subclass
                               do j = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                                  do l = 1,nitem !  Distribute over all item types
                                     do iw = 1, nw2 !  Disbribute over all weight increments
                                        actwgt(iw) = actwgt(iw) + bdols(iw,j,l,k)
                                        sum = sum + bdols(iw,j,l,k)
                                     end do
                                  end do
                               end do
                            end if
                         end do
                         if (sum.gt.0.) then
                            do iw = 1, nw2
                               bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                            bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                            end do
                            bdols(nw,imod,iitem,iact) = 0.0
                         else
                            print*,'unable to distribute no weight for b, ',
     &                         imod,' act = ',acodes(iact), ' cost = ', bdols(nw,imod,iitem,iact)
                         end if
                      end if
                   end if
                end if
             end if
          end do
       end do
    end do

c   Add in redistributed no weight identical/top piece item costs
    do iact = 1, nact2
       do imod = 1, nmod
          do iw = 1, nw
             do iitem = 1, nitem
                bdols(iw,imod,iitem,iact) = bdols(iw,imod,iitem,iact) + bdist(iw,imod,iitem,iact)
                bdist(iw,imod,iitem,iact) = 0.0
             end do
          end do
```

```
      end do
    end do

    Residual distribution of identical/top piece container no weight costs
    do iact = begmail, nact2
      do imod = 1, nmod
        do icon = 1, ncon
          if (fdols(nw,imod,icon,iact).gt.0) then
            sum = 0.0
            do iw = 1, nw2 !  Distribute over all weight increments
              sum = sum + fdols(iw,imod,icon,iact)
            end do
            if (sum.gt.0.0) then
              do iw = 1, nw2
                fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &            fdols(nw,imod,icon,iact)*fdols(iw,imod,icon,iact)/sum
              end do
              fdols(nw,imod,icon,iact) = 0.0
            end if
          end if
        end do
      end do
    end do

    Residual distribution of identical/top piece container no weight costs
    do iact = begmail, nact2
      do imod = 1, nmod
        do icon = 1, ncon
          if (fdols(nw,imod,icon,iact).gt.0.0) then
            sum = 0.0
            check = 0.0
            do iw = 1, nw2
              actwgt(iw) = 0.0
            end do
            do j = 1, nmod2 !  Distbribute over all cost pools (exclude LDC 15 proxy (pool #43))
              do iw = 1, nw2 !  Distribute over all weight increments
                actwgt(iw) = actwgt(iw) + fdols(iw,j,icon,iact)
                sum = sum + fdols(iw,j,icon,iact)
              end do
            end do
            if (sum.gt.0) then
              do iw = 1, nw2
                fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &            fdols(nw,imod,icon,iact)*actwgt(iw)/sum
              end do
              fdols(nw,imod,icon,iact) = 0.0
            else
              if (fdols(nw,imod,icon,iact).gt.0.) then
                print*,'Level 3 distribution of f act = ',acodes(iact)
                do k = begmail, nact2
                  do iw = 1, nw2
                    actshr2(iw,k) = 0.
                  end do
                end do
                do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                  if (class(k).eq.class(iact)) then !  Same subclass
                    do iw = 1, nw2 !  Distribute over all weight increments
                      actshr2(iw,k) = actshr2(iw,k) + fdols(iw,imod,icon,k)
                      sum = sum + fdols(iw,imod,icon,k)
                    end do
                  end if
                end do
                if (sum.gt.0.) then
                  do k = begmail, nact2
                    do iw = 1, nw2
                      fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &                  fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                    end do
                  end do
                  fdols(nw,imod,icon,iact) = 0.0
                else
                  print*,'Level 4 distribution f of act = ',acodes(iact)
                  do k = begmail, nact2
                    do iw = 1, nw2
                      actshr2(iw,k) = 0.
                    end do
                  end do
                  do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                    if (class(k).eq.class(iact)) then !  Same subclass
                      do j = 1,nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
```

```
                                do iw = 1, nw2 !  Distribute over all weight increments
                                   actshr2(iw,k) = actshr2(iw,k) + fdols(iw,j,icon,iact)
                                   sum = sum + fdols(iw,j,icon,iact)
                                end do
                             end do
                          end if
                       end do
                       if (sum.gt.0.) then
                          do k = begmail, nact2
                             do iw = 1, nw2
                                fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &                             fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                             end do
                          end do
                          fdols(nw,imod,icon,iact) = 0.0
                       else
                          print*,'unable to distribute no weight f for ',
     &                       imod,' act = ',acodes(iact), ' cost = ', fdols(nw,imod,icon,iact)
                       end if
                    end if
                 end if
              end if
           end if
        end do
     end do
  end do

c    Add in redistributed no weight identical/top piece container costs
     do iact = 1, nact2
        do imod = 1, nmod
           do iw = 1, nw
              do icon = 1, ncon
                 fdols(iw,imod,icon,iact) = fdols(iw,imod,icon,iact) + fdist(iw,imod,icon,iact)
                 fdist(iw,imod,icon,iact) = 0.0
              end do
           end do
        end do
     end do

c    Distribute mixed/empty item costs ("D" matrix) using direct item costs ("B" matrix) as a distribution key
c    over all activity codes and weight increments within cost pool and item type

     print *,' distributing D '
     do imod = 1, nmod
        do iitem = 1, nitem
           if (ddols(imod,iitem).gt.0.) then
              sum = 0.
              do iact = 1, nact2 !  Distribute over all activity code
                 do iw = 1, nw !  Distribute over all weight increments
                    sum = sum + bdols(iw,imod,iitem,iact)
                 end do
              end do
              if (sum.gt.0) then
                 do iact = 1, nact2
                    do iw = 1, nw
                       cdist(iw,imod,iitem,iact) = cdist(iw,imod,iitem,iact) +
     &                    ddols(imod,iitem) * bdols(iw,imod,iitem,iact) / sum
                    end do
                 end do
                 ddols(imod,iitem) = 0.
              end if
           end if
        end do
     end do

c    Distribute remaining mixed/empty item costs ("D" matrix) over all activity codes, weight increments,
C    and cost pools within item type using direct item costs ("B" matrix)
     do iitem = 1, nitem
        do imod = 1, nmod
           if (ddols(imod,iitem).gt.0.) then
              print *,'residual distribution of item = ',iitem,' pool = ',imod
              sum = 0
              do iact = 1, nact2
                 do iw = 1, nw
                    actshr(iw,iact) = 0.
                 end do
              end do
              do iact = 1, nact2 !  Distribute over all activity codes
                 do j = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                    do iw = 1, nw !  Distribute over all weight increments
```

```
                  actshr(iw,iact) = actshr(iw,iact) + bdols(iw,j,iitem,iact)
                  sum = sum + bdols(iw,j,iitem,iact)
               end do
            end do
         end do
         if (sum.gt.0.) then
            do iact = 1, nact2
               do iw = 1, nw
                  cdist(iw,imod,iitem,iact) = cdist(iw,imod,iitem,iact) +
     &               ddols(imod,iitem) * actshr(iw,iact) / sum
               end do
            end do
         else
            print *,' unable to dist D dols for iitem = ',iitem,', ',ddols(imod,iitem)
         end if
      end if
   end do
end do

C     Sum distributed mixed/empty item costs in "C" matrix
      do iact = 1, nact2
         do iitem = 1, nitem
            do imod = 1, nmod
               do iw = 1, nw
                  cdols(iw,imod,iitem,iact) = cdols(iw,imod,iitem,iact) + cdist(iw,imod,iitem,iact)
                  cdist(iw,imod,iitem,iact) = 0.
               end do
            end do
         end do
      end do

C     Distribute "identified" container costs ("G" matrix)

      do imod = 1, nmod          !  Initial distribution within cost pools

         if (imod.ne.20) then    !  Excludes 1Platform

            do icsi = 1, ncsi
               if (icsi.le.5) then ! Loose shapes distributed based upon direct piece costs ("A" matrix)
                  sum = 0.
                  distsum = 0.
                  do iact = 1, nact2 !  Distribute over all activity codes
                     do iw = 1, nw !  Distribute over all weight increments
                        sum = sum + adols(iw,imod,iact,icsi)
                     end do
                  end do
                  if (sum.gt.0.) then
                     do icon = 1, ncon
                        if (gdols(imod,icon,icsi).gt.0.) then
                           do iact = 1, nact2
                              do iw = 1, nw
                                 gdist(iw,imod,icon,iact) =
     &                              gdist(iw,imod,icon,iact) +
     &                              gdols(imod,icon,icsi) *
     &                              adols(iw,imod,iact,icsi) / sum
                              end do
                           end do
                        end if
                     end do
                  else          ! distribute over all cost pools, activity codes, and weight increments
                     do iact = 1, nact2 !  Distribute over all activity codes
                        do i = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                           do iw = 1 , nw !  Distribute over all weight increments
                              distsum = distsum + adols(iw,i,iact,icsi)
                           end do
                        end do
                     end do
                     if (distsum.gt.0) then
                        do iact = 1, nact2
                           do icon = 1, ncon
                              do i = 1, nmod2
                                 do iw = 1, nw
                                    gdist(iw,imod,icon,iact) =
     &                                 gdist(iw,imod,icon,iact) +
     &                                 gdols(imod,icon,icsi) *
     &                                 adols(iw,i,iact,icsi)/distsum
                                 end do
                              end do
                           end do
                        end do
```

```
            else       !  Undistributed costs included with uncounted/empty containers ("H" matrix)
               print *,'shape distribution empty: mod = ',imod,
&                 ', shape = ',icsi
               do icon = 1, ncon
                  hdols(imod,icon) = hdols(imod,icon) +
                     gdols(imod,icon,icsi)
               end do
            end if
         end if
      else             ! Items distributed upon direct item costs ("B" matrix)
         iitem = icsi - nshp2  !  Distribute over all item types
         sum = 0.
         distsum = 0.
         do iact = 1, nact2 !  Distribute over all activity codes
            do iw = 1, nw ! Distribute over all weight increments
               sum = sum + bdols(iw,imod,iitem,iact)
            end do
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do iact = 1, nact2
                     do iw = 1, nw
                        gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
&                          gdols(imod,icon,icsi) * bdols(iw,imod,iitem,iact) / sum
                     end do
                  end do
               end if
            end do
         else          ! distribute over all cost pools, activity codes, and weight increments
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + bdols(iw,i,iitem,iact)
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
                     do i = 1, nmod2
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) =
&                             gdist(iw,imod,icon,iact) +
&                             gdols(imod,icon,icsi) *
&                             bdols(iw,i,iitem,iact)/distsum
                        end do
                     end do
                  end do
               end do
            else       !  Undistributed costs included with uncounted/empty containers ("H" matrix)
               print *,'shape distribution empty: mod = ',imod,
&                 ', shape = ',icsi
               do icon = 1, ncon
                  hdols(imod,icon) = hdols(imod,icon) +
&                    gdols(imod,icon,icsi)
               end do
            end if
         end if
      end if

   end do

else if (imod.eq.20) then ! Distribute Platform over all allied labor cost pools, activity codes, and weight increments

   do icsi = 1, ncsi
      if (icsi.le.5) then ! Loose shapes distributed based upon direct piece costs ("A" matrix)
         sum = 0.
         do i = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
            do iact = 1, nact2 !  Distribute over all activity codes
               do iw = 1, nw !  Distribute over all weight increments
                  if ((i.eq.10).or.((i.ge.16).and.(i.le.23))) then ! Distribute over allied labor cost pools
                     sum = sum + adols(iw,i,iact,icsi)
                  end if
               end do
            end do
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
```

```
                do i = 1, nmod2
                   do iact = 1, nact2
                      do iw = 1, nw
                         if ((i.eq.10).or.((i.ge.16).and.(i.le.23))) then ! *CHECK this should be allied labor mod groups
                            gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
                               gdols(imod,icon,icsi) * adols(iw,i,iact,icsi) / sum
                         end if
                      end do
                   end do
                end do
             end if
          end do
       else          ! distribute over all cost pools, activity codes, and weight increments
          print *,'platform level 2 shape = ',icsi
          do iact = 1, nact2 !  Distribute over all activity codes
             do i = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                do iw = 1 , nw !  Distribute over all weight increments
                   distsum = distsum + adols(iw,i,iact,icsi)
                end do
             end do
          end do
          if (distsum.gt.0) then
             do iact = 1, nact2
                do icon = 1, ncon
                   do i = 1, nmod2
                      do iw = 1, nw
                         gdist(iw,imod,icon,iact) =
&                           gdist(iw,imod,icon,iact) +
&                           gdols(imod,icon,icsi) *
&                           adols(iw,i,iact,icsi)/distsum
                      end do
                   end do
                end do
             end do
          else       !  Undistributed Platform costs included with uncounted/empty containers ("H" matrix)
             print *,'shape distribution empty: mod = ',imod,
&               ', shape = ',icsi
             do icon = 1, ncon
                hdols(imod,icon) = hdols(imod,icon) +
                   gdols(imod,icon,icsi)
             end do
          end if
       end if
    else             ! Platform items distributed upon identical/top piece item costs ("B" matrix)
       iitem = icsi - nshp2
       sum = 0.
       do i = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
          do iact = 1, nact2 !  Distribute over all activity codes
             do iw = 1, nw !  Distribute over all weight increments
                if ((i.eq.10).or.((i.ge.16).and.(i.le.23))) then ! Distribute over allied labor cost pools
                   sum = sum + bdols(iw,i,iitem,iact)
                end if
             end do
          end do
       end do
       if (sum.gt.0.) then
          do icon = 1, ncon
             if (gdols(imod,icon,icsi).gt.0.) then
                do i = 1, nmod2
                   do iact = 1, nact2
                      do iw = 1, nw
                         if ((i.eq.10).or.((i.ge.16).and.(i.le.23))) then ! *CHECK this should be allied labor mod groups
                            gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
&                              gdols(imod,icon,icsi) * bdols(iw,i,iitem,iact) / sum
                         end if
                      end do
                   end do
                end do
             end if
          end do
       else          ! distribute over all cost pools, activity codes, and weight increments
          print *,'platform level 2 item = ',iitem
          do iact = 1, nact2 !  Distribute over all activity codes
             do i = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                do iw = 1 , nw !  Distribute over all weight increments
                   distsum = distsum + bdols(iw,i,iitem,iact)
                end do
             end do
          end do
          if (distsum.gt.0) then
```

```
                  do iact = 1, nact2
                     do icon = 1, ncon
                        do i = 1, nmod2
                           do iw = 1, nw
                              gdist(iw,imod,icon,iact) =
                                 gdist(iw,imod,icon,iact) +
     &                           gdols(imod,icon,icsi) *
     &                           bdols(iw,i,iitem,iact)/distsum
                              check = check + gdols(imod,icon,icsi)*
     &                           bdols(iw,i,iitem,iact)/distsum
                           end do
                        end do
                     end do
                  end do
               else       !  Undistributed Platform costs included with uncounted/empty containers ("H" matrix)
                  print *,'item distribution empty: mod = ',imod,
     &               ', item = ',icsi
                  do icon = 1, ncon
                     hdols(imod,icon) = hdols(imod,icon) +
     &                  gdols(imod,icon,icsi)
                  end do
               end if
            end if
         end if

      end do
   end if
end do                    ! End of "identified" container ("G" matrix) distribution

!  Sum distributed "identified" container costs into direct container costs ("F" matrix)
do iact = 1, nact2
   do icon = 1, ncon
      do imod = 1, nmod
         do iw = 1, nw
            fdols(iw,imod,icon,iact) = fdols(iw,imod,icon,iact) + gdist(iw,imod,icon,iact)
            gdist(iw,imod,icon,iact) = 0.
         end do
      end do
   end do
end do

c  Distribute uncounted/empty containers ("H" matrix) using direct and distributed "identified"
C  container costs ("F" matrix) over all activity codes and weight increments within cost pool and
C  container type
   do imod = 1, nmod
      do icon = 1, ncon
         sum = 0.
         do iact = 1, nact2  !  Distribute over all activity codes
            do iw = 1, nw    !  Distribute over all weight increments
               sum = sum + fdols(iw,imod,icon,iact)
            end do
         end do
         if (sum.gt.0) then
            do iact = 1, nact2
               do iw = 1, nw
                  gdist(iw,imod,icon,iact) =
     &               hdols(imod,icon) * fdols(iw,imod,icon,iact) / sum
               end do
            end do
            hdols(imod,icon) = 0.
         end if
      end do
   end do

c  Distribute remaining uncounted/empty container costs ("H" matrix) over all activity codes, weight
C  increments, and cost pools within container type using direct/distributed "identified" container
C  costs ("F" matrix)

   do icon = 1, ncon
      do imod = 1, nmod
         if (hdols(imod,icon).gt.0.) then
            sum = 0.
            do iact = 1, nact2
               do iw = 1, nw
                  actshr(iw,iact) = 0.
               end do
            end do
            do iact = 1, nact2 !  Distribute over all activity codes
               do j = 1, nmod2 !  Distribute over all cost pools (exclude LDC 15 proxy (pool #43))
                  do iw = 1, nw !  Distribute over all weight increments
```

```
                  actshr(iw,iact) = actshr(iw,iact) + fdols(iw,j,icon,iact)
                  sum = sum + fdols(iw,j,icon,iact)
               end do
            end do
         end do
         if (sum.gt.0.) then
            do iact = 1, nact2
               do iw = 1, nw
                  gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
     &               actshr(iw,iact)/sum * hdols(imod,icon)
               end do
            end do
         else
            print *,' unable to dist h dols for imod = ',imod,
     &         ' icon = ',icon
         end if
      end if
   end do
 end do

c    Sum up all costs (direct and redistributed) except not handling costs ("J" matrix)
c    Pieces
      do ishp = 1, nshp
         do iact = 1, nact2
            do imod = 1, nmod2
               do iw = 1, nw
                  result(iw,imod,iact) = result(iw,imod,iact) + adols(iw,imod,iact,ishp)
                  resulta(iw,imod,iact) = resulta(iw,imod,iact) + adols(iw,imod,iact,ishp)
               end do
            end do
         end do
      end do
c     Items
      do iact = 1, nact2
         do iitem = 1, nitem
            do imod = 1, nmod2
               do iw = 1, nw
                  result(iw,imod,iact) = result(iw,imod,iact) + bdols(iw,imod,iitem,iact)
                     + cdols(iw,imod,iitem,iact)
                  resultb(iw,imod,iact) = resultb(iw,imod,iact) + bdols(iw,imod,iitem,iact)
     &               + cdols(iw,imod,iitem,iact)
               end do
            end do
         end do
      end do
c     Containers
      do iact = 1, nact2
         do icon = 1, ncon
            do imod = 1, nmod2
               do iw = 1, nw
                  result(iw,imod,iact) = result(iw,imod,iact) + fdols(iw,imod,icon,iact) +
     &               gdist(iw,imod,icon,iact)
                  resultf(iw,imod,iact) = resultf(iw,imod,iact) + fdols(iw,imod,icon,iact) +
     &               gdist(iw,imod,icon,iact)
               end do
            end do
         end do
      end do

C     Distribute not handling costs ("J" matrix) using all other costs ("results" matrix)
      do imod = 1, nmod2        ! All cost pools except LDC 15 proxy
         sum = 0.
         distsum = 0.
c    Exclude allied cost pools (including 1Sacks_M, excluding 1CancMPP), 1EEqmt, Support Fcn 1,
C    and Support Fcn 4
         if (((imod.le.15).and.(imod.ne.10)).or.(imod.eq.17).or.((imod.ge.24).and.(imod.le.28)).or.
     &      ((imod.ge.32).and.(imod.le.37)).or.((imod.ge.40).and.(imod.le.43))) then
            do iact = 1, nact2  !  Distribute over all activity codes
               do iw = 1, nw    !  Distribute over all weight increments
                  sum = sum + result(iw,imod,iact)
               end do
            end do
            if (sum.gt.0) then
               do iact = 1, nact2
                  do iw = 1, nw
                     work(iw,imod,iact) = work(iw,imod,iact) +
     &                  jdols(imod) * result(iw,imod,iact) / sum
                  end do
               end do
            else
```

```
            print *,' unable to distribute J dollars for ',imod
         end if

   Allied not-handling (except cancellation) is distributed across LDCs 11-19, 79
   except Registry, BusReply, & Support Fcn 1
      else if (((imod.ge.16).and.(imod.le.23)).and.(imod.ne.17).or.(imod.eq.10)) then !  Allied cost pools except cancellation
         do iact = 1, nact2  !  Distribute over all activity codes
            do i = 1, nmod2  !  Distribute over all LDC 11-14, 17 cost pools
               if ((((ldc1(i).ge.11).and.(ldc1(i).le.19)).or.(ldc1(i).eq.79))
     &             .and.((i.ne.27).and.(i.ne.24).and.(i.ne.30).and.(i.ne.31))) then
                  do iw = 1, nw !  Distribute over all weight increments
                     distsum = distsum + result(iw,i,iact)
                  end do
               end if
            end do
         end do
         if (distsum.gt.0) then
            do iact = 1, nact2
               do i = 1, nmod2
                  if ((((ldc1(i).ge.11).and.(ldc1(i).le.19)).or.(ldc1(i).eq.79))
     &                .and.((i.ne.27).and.(i.ne.24).and.(i.ne.30).and.(i.ne.31))) then
                     do iw = 1, nw
                        work(iw,imod,iact) = work(iw,imod,iact) +
     &                     jdols(imod) * result(iw,i,iact) / distsum
                     end do
                  end if
               end do
            end do
         else
            print *,' unable to distribute J dollars for ',imod
         end if

c     1EEQMT is distributed across all MODS cost pools; distribution includes special services
C     exclude Support Fcn 1 & 4, Registry, BusReply, and LD48 SSV cost pools
      else if (imod.eq.29) then !  1EEqmt
         do iact = 1, nact2  !  Distribute over all activity codes
            do i = 1, nmod2  !  Distribute over all cost pools as noted above
               if ((i.ne.38).and.(i.ne.39).and.(i.ne.30).and.(i.ne.31)
     &             .and.(i.ne.27).and.(i.ne.24).and.(i.ne.40)) then
                  do iw = 1, nw !  Distribute over all weight increments
                     distsum = distsum + result(iw,i,iact)
                  end do
               end if
            end do
         end do
         if (distsum.gt.0) then
            do iact = 1, nact2
               do i = 1, nmod2
                  if ((i.ne.38).and.(i.ne.39).and.(i.ne.30).and.(i.ne.31)
     &                .and.(i.ne.27).and.(i.ne.24).and.(i.ne.40)) then
                     do iw = 1, nw
                        work(iw,imod,iact) = work(iw,imod,iact) +
     &                     jdols(imod) * result(iw,i,iact) / distsum
                     end do
                  end if
               end do
            end do
         else
            print *,' unable to distribute J dollars for ',imod
         end if

c     1SUPPORT is distributed across all pools (including special service activity codes)
C     exclude Support Fcn 1 & 4, Registry, BusReply, and LD48 SSV cost pools

      else if (imod.eq.31) then !  Support Fcn 1 (not including Misc)
         do iact = 1, nact2  !  Distribute over all activity codes
            do i = 1, nmod2  !  Distribute over all cost pools except as noted above
               if ((i.ne.38).and.(i.ne.39).and.(i.ne.30).and.(i.ne.31)
     &             .and.(i.ne.27).and.(i.ne.24).and.(i.ne.40)) then
                  do iw = 1, nw !  Distribute over all weight increments
                     distsum = distsum + result(iw,i,iact)
                  end do
               end if
            end do
         end do
         if (distsum.gt.0) then
            do iact = 1, nact2
               do i = 1, nmod2
                  if ((i.ne.38).and.(i.ne.39).and.(i.ne.30).and.(i.ne.31)
     &                .and.(i.ne.27).and.(i.ne.24).and.(i.ne.40)) then
```

```
                  do iw = 1, nw
                     work(iw,imod,iact) = work(iw,imod,iact) +
&                       jdols(imod) * result(iw,i,iact) / distsum
                  end do
               end if
            end do
         end do
      else
         print *,' unable to distribute J dollars for ',imod
      end if

:  Support Fcn 4 Admin/Other distributed over Function 4 cost pools, except LD48 Exp
   else if ((imod.eq.38).or.(imod.eq.39)) then !  LD48 Oth, LD48 Adm
      do iact = 1, nact2  !  Distribute over all activity codes
         do i = 1, nmod   !  Distribute over all Function 4 cost pools
            if ((ldc1(i).ge.41).and.(ldc1(i).le.48).and.(i.ne.37)) then ! Exclude LD48 Exp
               do iw = 1, nw !  Distribute over all weight increments
                  distsum = distsum + result(iw,i,iact)
               end do
            end if
         end do
      end do
      if (distsum.gt.0) then
         do iact = 1, nact2
            do i = 1, nmod
               if ((ldc1(i).ge.41).and.(ldc1(i).le.48).and.(i.ne.37)) then
                  do iw = 1, nw
                     work(iw,imod,iact) = work(iw,imod,iact) +
&                       jdols(imod) * result(iw,i,iact) / distsum
                  end do
               end if
            end do
         end do
      else
         print *,' unable to distribute J dollars for ',imod
      end if
   end if

   end do

c  Sum distributed not handling costs ("J" matrix) into handling costs ("results" matrix)
   print *,' summing J into all '
   do iact = 1, nact2
      do imod = 1, nmod2
         do iw = 1, nw
            result(iw,imod,iact) = result(iw,imod,iact) + work(iw,imod,iact)
            resultj(iw,imod,iact) = work(iw,imod,iact)
            work(iw,imod,iact) = 0.
         end do
      end do
   end do

c  Redistribute class-specific mixed mail costs over appropriate class-specific direct activity codes,
c  weight increment, and within cost pools
   do imod = 1,nmod
      do iact = 1,nmixcl
         do iw = 1, nw
            if (result(iw,imod,nact+iact).gt.0.0) then
               sum = 0.
               do i = 1,nact
                  actshr3(i) = 0.
               end do
               do i = 1,nact !  Distribute over all direct activity codes
                  do j = 1,nw !  Distribute over all weight increments
                     if (mixmap(i,iact).gt.0) then
                        sum = sum + result(j,imod,mixmap(i,iact))
                        actshr3(mixmap(i,iact)) = actshr3(mixmap(i,iact))
&                          + result(j,imod,mixmap(i,iact))
                     end if
                  end do
               end do
               if (sum.gt.0.) then
                  print*, 'Sum = ', sum, ' for actv ', acodes(nact+iact)
                  do i = 1,nact
                     if (mixmap(i,iact).gt.0) then
                        work(iw,imod,mixmap(i,iact)) =
&                          work(iw,imod,mixmap(i,iact)) +
&                          (result(iw,imod,nact+iact)*
&                          actshr3(mixmap(i,iact))/sum)
                     end if
```

```
                  end do
                  result(iw,imod,nact+iact) = 0.
               else          !  Distribute over all cost pools
                  print*, 'Residual for mix actv code ', acodes(nact+iact)
                  sum = 0.
                  do i = 1,nact
                     actshr3(i) = 0.
                  end do
                  do i = 1, nact !  Distribute over all direct activity codes
                     do j = 1,nw !  Distribute over all weight increments
                        do k = 1, nmod !  Distribute over all cost pools
                           if (mixmap(i,iact).gt.0) then
                              sum = sum + result(j,k,mixmap(i,iact))
                              actshr3(mixmap(i,iact)) = actshr3(mixmap(i,iact))
     &                           + result(j,k,mixmap(i,iact))
                           end if
                        end do
                     end do
                  end do
                  if (sum.gt.0.) then
                     do i = 1, nact
                        if (mixmap(i,iact).gt.0) then
                           work(iw,imod,mixmap(i,iact)) =
     &                        work(iw,imod,mixmap(i,iact)) +
     &                        (result(iw,imod,nact+iact)*
     &                        actshr3(mixmap(i,iact))/sum)
                        end if
                     end do
                     result(iw,imod,nact+iact) = 0.
                  else
                     print*, 'Mix actv code not distributed ', acodes(nact+iact),
     &                  ' cost = ', result(iw,imod,nact+iact), ' pool ', modcodes(imod)
                  end if
               end if
            end if
         end do
      end do
   end do

c  Sum distributed class-specific mixed-mail costs into all other costs
   do iact = 1, nact
      do imod = 1, nmod
         do iw = 1, nw
            result(iw,imod,iact) = result(iw,imod,iact) + work(iw,imod,iact)
            work(iw,imod,iact) = 0.
         end do
      end do
   end do

C  Replace LDC 15 w/proxy distribution key
   do iact = 1, nact
      do iw = 1, nw
         result(iw,15,iact) = result(iw,nmod,iact)
      end do
   end do

c  Compute volume-variable costs for all cost pools except Support Fcn 1 & 4
   distsum = 0.
   distsum48 = 0.
   do imod = 1,nmod
      if ((ldc1(imod).ge.11).and.(ldc1(imod).le.19).and.(imod.ne.31)) then
         distsum = distsum + pooldols(imod) !  Function 1 costs
      end if
      if ((ldc1(imod).ge.41).and.(ldc1(imod).le.49).and.(imod.ne.39)) then !
         distsum48 = distsum48 + pooldols(imod) !  Function 4 costs
      end if
      sum = 0.
      do iact = 1,nact
         do iw = 1,nw
            sum = sum + result(iw,imod,iact)
         end do
      end do
      if (sum.gt.0.) then
         do iact = 1,nact     !  Distribute over all direct activity codes
            do iw = 1,nw      !  Distribute over all weight increments
               if ((imod.ne.31).and.(imod.ne.30).and.(imod.ne.38).and.(imod.ne.39)) then
                  varcost(iw,imod,iact) = varcost(iw,imod,iact) +
     &                pooldols(imod)*variable(imod)*result(iw,imod,iact)/sum !  All
                  if (((ldc1(imod).ge.11).and.(ldc1(imod).le.19)).or.(ldc1(imod).eq.79)) then
                     work(iw,31,iact) = work(iw,31,iact) + varcost(iw,imod,iact) !  Distrib key for Support Fcn 1
```

```
                else if ((ldc1(imod).ge.41).and.(ldc1(imod).le.49)) then
                   work(iw,39,iact) = work(iw,39,iact) + varcost(iw,imod,iact) !  Distrib key for Support Fcn 4
                end if
                novarcst(iw,imod,iact) = novarcst(iw,imod,iact) +
                   pooldols(imod)*result(iw,imod,iact)/sum
             end if
          end do
       end do
    else
       print *,'unable to distribute $ = ',pooldols(imod),
&         ' for mods pool ',modcodes(imod)
    end if
 end do

 distsum48 = distsum48 + 765750. !  Add in Window Service cost pool dollars

 Uses windows cost pool $ as dist key (MODS % of total (MODS,Non-MODS,BMC)
 MODS, Non-MODS, BMCS cost pool $ (denominator) to add in CRA window service costs (MODS portion)

 do iact = 1, nact
    do iw = 1, nw
       work(iw,39,iact) = work(iw,39,iact) + wincost(iact,iw)*
&         765750./(765750.+1400378.+341.)
    end do
 end do

 Function 1 support piggybacked on other Function 1 mail processing
 Calculation of volume-variable costs for Support Fcn 1 & 4 cost pools
 sum = 0.
 sum48 = 0.
 do iact = 1,nact
    do iw = 1,nw
       sum = sum + work(iw,31,iact)
       sum48 = sum48 + work(iw,39,iact)
    end do
 end do
 if ((sum.gt.0.).and.(sum48.gt.0.)) then
    do iact = 1,nact
       do iw = 1,nw
          varcost(iw,31,iact) = varcost(iw,31,iact) +
&            pooldols(31)*variable(31)*work(iw,31,iact)/sum
          novarcst(iw,31,iact) = novarcst(iw,31,iact) +
&            pooldols(31)*work(iw,31,iact)/sum
          varcost(iw,30,iact) = varcost(iw,30,iact) +
&            pooldols(30)*variable(30)*work(iw,31,iact)/sum
          novarcst(iw,30,iact) = novarcst(iw,30,iact) +
&            pooldols(30)*work(iw,31,iact)/sum
          varcost(iw,39,iact) = varcost(iw,39,iact) +
&            pooldols(39)*variable(39)*work(iw,39,iact)/sum48
          novarcst(iw,39,iact) = novarcst(iw,39,iact) +
&            pooldols(39)*work(iw,39,iact)/sum48
          varcost(iw,38,iact) = varcost(iw,38,iact) +
&            pooldols(38)*variable(38)*work(iw,39,iact)/sum48
          novarcst(iw,38,iact) = novarcst(iw,38,iact) +
&            pooldols(38)*work(iw,39,iact)/sum48
       end do
    end do
 else if (sum.eq.0.) then
    print *,'unable to distribute $ = ',pooldols(31),
&      ' for mods pool ',modcodes(31)
 else if (sum48.eq.0.) then
    print *,'unable to distribute $ = ',pooldols(39),
&      ' for mods pool ',modcodes(39)
 end if

 sum=sum/distsum
 sum48=sum48/distsum48

 print*,'1Support variability = ',sum
 print*,'LD48 A/O variability = ',sum48

 open(80,file='mods002by.data')
81 format(i3,i4,i3,8f18.9)

 do imod = 1, nmod2
    do iact = 1, nact
       do iw = 1, nw
          write (80,81) ldc1(imod), iact, iw, varcost(iw,imod,iact),
&            novarcst(iw,imod,iact), result(iw,imod,iact),
&            resulta(iw,imod,iact), resultb(iw,imod,iact),
```

```
&            resultf(iw,imod,iact), resultj(iw,imod,iact),work(iw,imod,iact)
       end do
     end do
  end do

   rint *, ' Total Count and Dollars by Matrix '
   rite (*,'(2x,a1,i6,f15.2)') 'A', acnt, atot
  write (*,'(2x,a1,i6,f15.2)') 'B', bcnt, btot
  write (*,'(2x,a1,i6,f15.2)') 'C', ccnt, ctot
  write (*,'(2x,a1,i6,f15.2)') 'D', dcnt, dtot
  write (*,'(2x,a1,i6,f15.2)') 'F', fcnt, ftot
  write (*,'(2x,a1,i6,f15.2)') 'G', gcnt, gtot
  write (*,'(2x,a1,i6,f15.2)') 'H', hcnt, htot
  write (*,'(2x,a1,i6,f15.2)') 'J', jcnt, jtot

  end

------------------------------------------------------------------------------------------
  Assigns shape
  function shapeind(actv,f9635,f9805)

  integer*4 shapeind, actv
  character*1  f9635
  character*4 f9805

  if (((actv.ge.1000).and.(actv.lt.2000)).or.(actv.eq.5431).or.(actv.eq.5441)
 &   .or.(actv.eq.5451).or.(actv.eq.5461)) then
     if (((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K')) then
        shapeind = 1          ! cards
     else
        shapeind = 2          ! letters
     end if
  else if (((actv.ge.2000).and.(actv.lt.3000)).or.(actv.eq.5432).or.(actv.eq.5442)
 &      .or.(actv.eq.5452).or.(actv.eq.5462)) then
     shapeind = 3             ! flats
  else if (((actv.ge.3000).and.(actv.lt.4000)).or.(actv.eq.5433).or.(actv.eq.5443)
 &      .or.(actv.eq.5453).or.(actv.eq.5463)) then
     shapeind = 4             ! IPPs
  else if (((actv.ge.4000).and.(actv.lt.5000)).or.(actv.eq.5434).or.(actv.eq.5444)
        .or.(actv.eq.5454).or.(actv.eq.5464)) then
     shapeind = 5             ! parcels
  else
     shapeind = 6             ! other?
  end if

  if (actv.eq.5340) then
     shapeind = 6  !  other?
     if (((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K')) then
        shapeind = 1          ! cards
     end if
     if (f9635.eq.'A') then
        shapeind = 2  !  letters
     end if
     if ((f9635.eq.'D').or.(f9635.eq.'E')) then
        shapeind = 3  !  flats
     end if
     if ((f9635.eq.'F').or.(f9635.eq.'G').or.(f9635.eq.'J')) then
        shapeind = 4  !  IPPs
     end if
     if ((f9635.eq.'H').or.(f9635.eq.'I')) then
        shapeind = 5  !  parcels
     end if
  end if

  if ((actv.ge.10).and.(actv.lt.1000)) then
     if ((f9805(1:1).eq.'1').and.(((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K'))) then
        shapeind = 1  !  cards
     else if (f9805(1:1).eq.'1') then
        shapeind = 2  !  letters
     else if (f9805(1:1).eq.'2') then
        shapeind = 3  !  flats
     else if (f9805(1:1).eq.'3') then
        shapeind = 4  !  IPPs
     else if (f9805(1:1).eq.'4') then
        shapeind = 5  !  parcels
     else
        shapeind = 6  !  other?
     end if
  end if
```

```
return
end
```

```
  signs weight increment

function weight(f165,if166,if167,ct_nowgt,nw)

character*1  f165
integer*4    if166, if167, weight, ct_nowgt, nw

weight = 0

if (f165.eq.'A') then
   weight = 1              !  < 1/2 ounce
else if (f165.eq.'B') then
   weight = 2              !  1 ounces
else if (f165.eq.'C') then
   weight = 3              !  1 1/2 ounces
else if (f165.eq.'D') then
   weight = 4              !  2 ounces
else if (f165.eq.'E') then
   weight = 5              !  2 1/2 ounces
else if (f165.eq.'F') then
   weight = 6              !  3 ounces
else if (f165.eq.'G') then
   weight = 7              !  3 1/2 ounces
else if (f165.eq.'H') then
   weight = 8              !  4 ounces
else if (f165.eq.'I') then
   if (if166.eq.0) then    !  < 1 lb
      if (if167.gt.0) then
         weight = if167 + 4
      else
         weight = nw
         ct_nowgt = ct_nowgt + 1
      end if
   else if ((if166.eq.1).and.(if167.eq.0)) then
      weight = 20
   else if ((if166.gt.1).or.((if166.eq.1).and.(if167.gt.0))) then
      weight = 21
   else
      weight = nw
      ct_nowgt = ct_nowgt + 1
   end if
else
   weight = nw
   ct_nowgt = ct_nowgt + 1
end if

return
end
```

```
      program sumclass_mod_wgt

      Purpose:  Sum distributed volume-variable mail processing costs for MODS 1&2 offices to subclass
                Costs are calculated in the Fortran program modsproc00_wgt.f

       mplicit none

      integer*4 nact, ncl, nmod, nshp, nmat, nshp2, nw

      parameter (nmod = 42)      !  Number of cost pools
      parameter (nact = 255)     !  Number of activity codes
      parameter (ncl = 60)       !  Number of subclasses
      parameter (nshp = 3)       !  Number of shapes
      parameter (nmat = 1)       !  Number of cost categories
      parameter (nshp2 = 5)      !  Number of shapes (class map)
      parameter (nw = 22)        !  Number of weight increments

      real*8    dollars(nmat,nw,nmod,nact)
      real*8   cdols(nmat,nmod,ncl,nshp,nw)

      integer*4 imod, iact, icl, i, j, k, shape, is
      integer*4 ier, shp(nact), iw, imod2
      integer*4 clmap(nact), mod(nmod), ldc1(nmod)

      character*16 grp(nmod)
      character*9  class(ncl), clcode
      character*9  class2(ncl)
      character*4  acodes(nact), temp, acin(nshp2)
      character*5  shapetype(nshp)/'1Ltr ','2Flt ','3Pcl '/

      ier = 0

      Map of cost pools
      open(30,file='costpools.00.619')
2     format(i2,a16,i2)

      do i = 1, nmod
         read(30,32) mod(i), grp(i), ldc1(i)
       end do
       rint *, 'Mod groups read'
      close(30)

      Map of activity codes
      open(20,file='activity00.ecr.cra')
1     format(a4)

      do i = 1, nact
         read (20,21) acodes(i)
         is = shape(acodes(i))  !  Assign shape
         shp(i) = is
      end do
      print*, 'Read in activity codes '
      close(20)

      Map of subclasses
      open(33,file='classes_intl.cra.new')
4     format(a9)
      do i = 1, ncl
         read(33,34) class(i)
         class2(i) = class(i)
      end do
      close(33)
      print*, 'Read in classes '

      Maps activity codes to subclass
      open(35,file='classmap_intl.new')
6     format(a9,5(4x,a4))
      do i = 1, nact
         clmap(i) =  0
      end do
      do while (ier.eq.0)
         read(35,36,iostat=ier,end=101) clcode, acin
         do i = 1, nshp2
            j = 0
            if (acin(i).ne.'    ') then
               do iact = 1,nact
                  if (acodes(iact).eq.acin(i)) then
                     j = iact
                  end if
               end do
```

```
            if (j.gt.0) then
               temp = acin(i)
               if ((temp(2:2).eq.'6').or.(temp(2:2).eq.'7').or.
     &             (temp(2:2).eq.'8').or.(temp(1:2).eq.'54')) then
                  clmap(j) = 17
               else
                  k = 0
                  do icl = 1,ncl
                     if (class2(icl).eq.clcode) then
                        k=icl
                     end if
                  end do
                  if (k.gt.0) then
                     clmap(j) = k
                  else
                     print *,' bad class code = ',clcode,' ',clcode
                  end if
               end if
            else
               print *,' activity code not found ',acin(i)
            end if
         end if
      end do
   end do
01 print *,' read exit of classmap = ',ier
   ier = 0
   close(35)

!  Initialize matrices
   do imod = 1, nmod
      do icl = 1, ncl
         do j = 1, nmat
            do is = 1, nshp
               do iw = 1, nw
                  cdols(j,imod,icl,is,iw) = 0.
               end do
            end do
         end do
      end do
   nd do

!     Read in distributed cost data
   open(40,file='mods002by.data')
11 format(10x,8f18.9)

   do imod = 1, nmod
      do iact = 1, nact
         do iw = 1, nw
            read (40,41) (dollars(j,iw,imod,iact),j=1,nmat)
         end do
      end do
   end do
   close(40)

C  Sum data to classes

   do j = 1, nmat
      do imod = 1, nmod
         do iact = 1, nact
            do iw = 1, nw
               icl = clmap(iact) !  Subclass for corresponding activity code
               is = shp(iact) !  Assign shape
               imod2 = imod
               if (imod2.eq.30) imod2 = 31 !  Combine Fcn 1 Support
               if (imod2.eq.39) imod2 = 38 !  Combine Fcn 4 Support
               if (icl.gt.0) then
                  cdols(j,imod2,icl,is,iw) = cdols(j,imod2,icl,is,iw)
     &               + dollars(j,iw,imod,iact)
               else
                  print *,' activity ',acodes(iact),' not in class map ', iact
               end if
            end do
         end do
      end do
   end do
   print*, 'Finished with data '

C     Write out costs by subclass, cost pool, shape, and weight increment

   grp(31) = 'Support Fcn1'
```

```
    grp(38) = 'Support Fcn4'

    open(55,file='mod00_wgt2.csv',recl=500)
    format(a9,',',i2,',',a16,',',i2,',','a5,',',22(f18.9,','))

     ) icl = 1, ncl
       do imod = 1, nmod
          do is = 1, nshp
             if ((imod.ne.30).and.(imod.ne.39)) then
                if ((icl.eq.1).or.(icl.eq.2).or.((icl.ge.8).and.(icl.le.11))) then
                   write(55,56) class(icl), imod, grp(imod), ldc1(imod),
    &                  shapetype(is), (cdols(1,imod,icl,is,iw), iw = 1, nw)
                end if
             end if
          end do
       end do
    end do

    end

------------------------------------------------------------------------------
    Assign shape

    function shape(act)

    integer*4    shape
    character*4  act

    if (act(1:1).eq.'1') then
       shape = 1  !  Letters
    else if (act(1:1).eq.'2') then
       shape = 2  !  Flats
    else if ((act(1:1).eq.'3').or.(act(1:1).eq.'4')) then
       shape = 3  !  IPPs/Parcels
    else
       shape = 3  !  Other (Special Service)
       if (act.gt.'1000') then
          print*, 'No shape for actv ', act
       end if
    end if

    return
    end
```

```
program bmcproc00_wgt

Purpose:  Computes distributed volume-variable costs (USPS Method) for BMCs
          Adds additional dimension for various weight categories

  plicit none

integer*4 nmod, nw, begmod, nw2
integer*4 nact, ishp, nshp, nmix, nmixcl, nact2
integer*4 nitem, nshp2, ncsi, ncon, begmail

parameter (nmod = 6)       !  Number of cost pools
parameter (begmod = 1)     !  Beginning position for BMC cost pools within map
parameter (nw =  22)       !  Number of weight increments (including no weight)
parameter (nw2 = 21)       !  Number of weight increments
parameter (nact = 255)     !  Number of direct activity codes
parameter (nshp = 6)       !  Number of shapes
parameter (nitem = 16)     !  Number of item types
parameter (nshp2 = 5)      !  Number of shapes (not including other)
parameter (ncon = 10)      !  Number of container types
parameter (nmix = 20)      !  Number of combined activity codes - for dist of counted items
parameter (ncsi = nshp2 + nitem) !  Number of "identified" container types (loose shapes + items)
parameter (begmail = 17)   ! Set this to the index of the first non-Spec Serv activity code
parameter (nmixcl = 20)    ! Number of class-specific mixed-mail codes
parameter (nact2 = 275)    ! Number of activity codes including class-specific mixed-mail

include 'iocs2000.h'

real*8     adols(nw,nmod,nact2,nshp) ! Handling direct single piece
real*8    adist(nw,nmod,nact2,nshp) ! Workspace for distribution of no weight single pieces
real*8     bdols(nw,nmod,nitem,nact2) ! Handling identical or top-piece item
real*8    bdist(nw,nmod,nitem,nact2) ! Workspace for distribution of no weight identical/top-piece items
real*8    cdist(nw,nmod,nitem,nact2) ! Workspace for distribution of matrix D
real*8    ddist(nw,nmod,nact2) ! Workspace for Level 3 D matrix distribution
real*8    dir9806(nw,nmod,nact2) ! Holds f9806 direct tallies matrix
real*8     cdols(nw,nmod,nitem,nact2) ! Workspace for distributed costs from matrix D
real*8     ddols(nmod,nitem) ! Handling mixed/empty item
real*8     fdols(nw,nmod,ncon,nact2) ! Handling identical or top-piece container
eal*8     fdist(nw,nmod,ncon,nact2) ! Workspace for distribution of no weight identical/top-piece containers
 eal*8     gdols(nmod,ncon,ncsi) ! Handling "identified" container
real*8    gdist(nw,nmod,ncon,nact2) ! G Matrix distributed to activity code
real*8    hdist(nw,nmod,ncon,nact2) ! H Matrix distributed to activity code
real*8    hkey(nw,nmod,ncon,nact2) ! Matrix to hold distribution key for unidentified container
real*8     hdols(nmod,ncon) ! Handling uncounted/empty container
real*8    result(nw,nmod,nact2) ! Array to hold results
real*8    result2(nw,nmod,nact2) ! Array to hold distribution key data for Matrix J
real*8    resulta(nw,nmod,nact2) ! Array to hold results for matrix A
real*8    resultb(nw,nmod,nact2) ! Array to hold results for matrix B, C, D
real*8    resultf(nw,nmod,nact2) ! Array to hold results for matrix F, G, H
real*8    resultj(nw,nmod,nact2) ! Array to hold distributed J matrix
real*8    work(nw,nmod,nact2) ! Array to hold distributed mixed class-specific
real*8    jdols(nmod)      ! Not Handling
real*8    counts(ncsi)
real*8    actshr(nw,nact2), actshr3(nact), actwgt(nw2), actshr2(nw,nact2)
real*8     dlrs, sum, distsum, rf9250, check
real*8    count1,count2
real*8    bmix(nmod,nact2)
real*8    atot, btot, ctot, dtot, ftot, gtot, htot, jtot
real*8    pooldols(nmod)
real*8    variable(nmod)
real*8    varcost(nw,nmod,nact)
real*8    dlrsin, dlrsout, dlrs5340in, dlrs5340out
real*8    novarcst(nw,nmod,nact)
real*8    nowgt_key(nw,nmod,nact2)

logical flag

integer*4 acnt, bcnt, ccnt, dcnt, fcnt, gcnt, hcnt, jcnt
integer*4 ind, ldc, if9806, iact2, l
integer*4 cnt,npl,npnl,counted
integer*4 i, j, imat, imod, icon, iact, icsi, iitem, shapeind, iw
integer*4 ier, k, class_code(nact2)
integer*4 mapcodes(20)
 nteger*4 searchc, searchi, modgrp, hand, actv
integer*4 mixcodes(nmixcl)
integer*4 acodes(nact2), mixcount (nmixcl)
integer*4 mixmap(nact,nmixcl)
integer*4 ldcl(nmod), class(nact2)
integer*4 modclass, pool, poolcode(nmod)
integer*4 if166, if167, weight, ct_nowgt
```

```
      character*14 modcodes(nmod)
      character*1 codes(26)/'A','B','C','D','E','F','G','H','I','J','K',
     &   'L','M','N','O','P','Q','R','S','T','U','V',
     &   'W','X','Y','Z'/

      logical flag2

      atot = 0.0
      btot = 0.0
      ctot = 0.0
      dtot = 0.0
      ftot = 0.0
      gtot = 0.0
      htot = 0.0
      jtot = 0.0
      acnt = 0
      bcnt = 0
      ccnt = 0
      dcnt = 0
      fcnt = 0
      gcnt = 0
      hcnt = 0
      jcnt = 0
      cnt = 0
      ier = 0
      count1 = 0.0
      count2 = 0.0
      dlrsin = 0.0
      dlrsout = 0.0
      dlrs5340in = 0.0
      dlrs5340out = 0.0
      npl = 0
      npnl = 0
      counted =0

      do i = 1, nmod
         pooldols(i) = 0.0
         variable(i) = 0.0
      end do

      do i = 1, 20
         mapcodes(i) = 0
      end do

      do i = 1, nmixcl
         mixcodes(i) = 0
         mixcount(i) = 0
      end do

C     Map of activity codes
      open(20,file='activity00.ecr.cra')
21    format(i4,i6,i5)
      do i=1,nact2
         read (20,21) acodes(i), class(i), class_code(i)
      end do
      print *,'read activity map'
      close(20)

c     Map of class specific mixed-mail activity codes
      open(20,file='mixclass.intl')
      do i = 1,nmixcl
         read (20,21) mixcodes(i)
      end do
      print *,'read mixed item code list'
      close(20)

      do i = 1,nact
         do j = 1,nmixcl
            mixmap(i,j) = 0
         end do
      end do

c     Maps class specific mixed-mail activity codes to appropriate direct activity codes
      open(20,file='mxmail.intl.dat')
23    format(20i4)

      do while (ier.eq.0)
         read (20,23,iostat=ier,end=75) mapcodes
         i = searchi(mixcodes,nmixcl,mapcodes(1))
```

```
      if (i.gt.0) then
         flag = .true.
         ind = 1
         do while ((flag).and.(ind.lt.20))
            ind = ind + 1
            if (mapcodes(ind).gt.0) then
               j = searchi(acodes,nact,mapcodes(ind))
               if (j.gt.0) then
                  mixcount(i) = mixcount(i) + 1
                  mixmap(mixcount(i),i) = j
               else
                  print *,' Direct mail code did not map ',mapcodes(ind)
               end if
            else
               flag = .false.
            end if
         end do
      else
         print *,' Mixed mail code did not map ',mapcodes(1)
      end if
   end do
5  print *,' read mixed-mail map with exit code = ',ier
   close(20)

   Map of cost pool dollars and variabilities by cost pool
   open(20,file='costpools.00.bmc.619')
4  format(i4,a14,i5,f9.0,f7.2)
   do i = 1,nmod
      read(20,24) poolcode(i), modcodes(i), ldc1(i), pooldols(i), variable(i)
   end do
   close(20)

C  Initialize matrices

   do iw = 1,nw
      do imod = 1,nmod
         do iact = 1,nact
            varcost(iw,imod,iact) = 0.
            novarcst(iw,imod,iact) = 0.
         end do
      end do
   end do
   do ishp = 1, nshp
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               adols(iw,imod,iact,ishp) = 0.0
               adist(iw,imod,iact,ishp) = 0.0
            end do
         end do
      end do
   end do
   do iact = 1, nact2
      do iitem = 1, nitem
         do imod = 1, nmod
            do iw = 1, nw
               bdols(iw,imod,iitem,iact) = 0.
               bdist(iw,imod,iitem,iact) = 0.
               bmix(imod,iact) = 0.0
            end do
         end do
      end do
   end do
   do iact = 1, nact2
      do iitem = 1, nitem
         do imod = 1, nmod
            do iw = 1, nw
               cdist(iw,imod,iitem,iact) = 0.
            end do
         end do
      end do
   end do
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw
            ddist(iw,imod,iact) = 0.
            dir9806(iw,imod,iact) = 0.
         end do
      end do
   end do
```

```
do iact = 1, nact2
   do iitem = 1, nitem
      do imod = 1, nmod
         do iw=1,nw
            cdols(iw,imod,iitem,iact) = 0.
         end do
      end do
   end do
end do
do iitem = 1, nitem
   do imod = 1, nmod
      ddols(imod,iitem) = 0.
   end do
end do
do iact = 1, nact2
   do icon = 1, ncon
      do imod = 1, nmod
         do iw = 1, nw
            fdols(iw,imod,icon,iact) = 0.
            fdist(iw,imod,icon,iact) = 0.
            hkey(iw,imod,icon,iact) = 0.
         end do
      end do
   end do
end do
do icsi = 1, ncsi
   do icon = 1, ncon
      do imod = 1, nmod
         gdols(imod,icon,icsi) = 0.
      end do
   end do
end do
do iact = 1, nact2
   do icon = 1, ncon
      do imod = 1, nmod
         do iw = 1, nw
            gdist(iw,imod,icon,iact) = 0.
            hdist(iw,imod,icon,iact) = 0.
         end do
      end do
   end do
end do
do icon = 1, ncon
   do imod = 1, nmod
      hdols(imod,icon) = 0.
   end do
end do
do iact = 1, nact2
   do imod = 1, nmod
      do iw = 1, nw
         result(iw,imod,iact) = 0.
         result2(iw,imod,iact) = 0.
      end do
   end do
end do
do iact = 1, nact2
   do imod = 1, nmod
      do iw = 1, nw
         resulta(iw,imod,iact) = 0.
      end do
   end do
end do
do iact = 1, nact2
   do imod = 1, nmod
      do iw = 1, nw
         resultb(iw,imod,iact) = 0.
      end do
   end do
end do
do iact = 1, nact2
   do imod = 1, nmod
      do iw = 1, nw
         resultf(iw,imod,iact) = 0.
      end do
   end do
end do
do iact = 1, nact2
   do imod = 1, nmod
      do iw = 1, nw
         work(iw,imod,iact) = 0.
```

```
            resultj(iw,imod,iact) = 0.
            nowgt_key(iw,imod,iact) = 0.
          end do
        end do
      end do
      do imod = 1, nmod
        jdols(imod) = 0.
      end do
      print*, 'Matrices initialized '

      open(25,file='bmcs_mp00by_new.dat',recl=1200) !  BMC mail proc IOCS data
1     format(a1167,f15.5,i2,i2,i3,i5)
      cnt = 0
      ier = 0
      ct_nowgt = 0

      do while (ier.eq.0)

        read(25,31,iostat=ier,end=100) rec,dlrs,pool,ldc,iw,actv

        cnt = cnt + 1

        modgrp = searchi(poolcode,nmod,pool) !  Assign cost pool code

        if ((modgrp.lt.begmod).or.(modgrp.gt.nmod)) then
          goto 99              !  Exclude break tallies
        end if

        read(f9250,'(f10.0)') rf9250
        read(f9806,'(i4)') if9806
        read(f166,'(i2)') if166
        read(f167,'(i2)') if167

        dlrs = rf9250/100000.

c     Break out Std A ECR Saturation and High Density into separate activity codes
        if ((actv.eq.1311).or.(actv.eq.2311).or.(actv.eq.3311).or.(actv.eq.4311)) then !  Std A  WSH/WSS
          if (f9619.eq.'1') then !  WSS
            if (actv.eq.1311) actv = 1313
            if (actv.eq.2311) actv = 2313
            if (actv.eq.3311) actv = 3313
            if (actv.eq.4311) actv = 4313
          end if
        end if

        if ((actv.eq.1331).or.(actv.eq.2331).or.(actv.eq.3331).or.(actv.eq.4331)) then !  Std A NP WSH/WSS
          if (f9619.eq.'1') then !  WSS
            if (actv.eq.1331) actv = 1333
            if (actv.eq.2331) actv = 2333
            if (actv.eq.3331) actv = 3333
            if (actv.eq.4331) actv = 4333
          end if
        end if

c     Any "auto" ECR flats or parcels are assumed to be basic ECR
        if (actv.eq.2312) actv = 2310
        if (actv.eq.3312) actv = 3310
        if (actv.eq.4312) actv = 4310
        if (actv.eq.2332) actv = 2330
        if (actv.eq.3332) actv = 3330
        if (actv.eq.4332) actv = 4330

        ishp = shapeind(actv,f9635,f9805) !  Subroutine assigns shape
        iitem = searchc(codes,nitem,f9214) !  Assign item type
        icon = searchc(codes,ncon,f9219) !  Assign container type

        if (if9806.ge.1000) then
          if9806 = actv
        end if

        iact = searchi(acodes,nact2,actv) !  Activity codes
        iact2 = searchi(acodes,nact2,if9806) !  Activity codes

        dlrsin = dlrsin + dlrs
        modclass=1

c     Assign handling category
        if (((actv.ge.1000).and.(actv.le.4950)).or.((actv.ge.5300).and.(actv.le.5480))) then
          hand = 1             ! direct (non special services)
        else if (actv.lt.1000) then
```

```
      if (((f9805.ge.'1000').and.(f9805.le.'4950')).or.
     &    ((f9805(1:2).ge.'53').and.(f9805(1:2).le.'54'))) then
           hand = 1          ! direct (non special service handling)
        else if ((f9635.ge.'A').and.(f9635.le.'K')) then
           hand = 1          ! direct (special service handling)
        else if ((f9214.ge.'A').and.(f9214.le.'P')) then
           hand = 2          ! mixed item
        else if ((f9219.ge.'A').and.(f9219.le.'J')) then
           hand = 3          ! mixed container
        else
           hand = 4          !  not handling mail
        end if
      else if ((f9214.ge.'A').and.(f9214.le.'P')) then
        hand = 2             ! mixed item
      else if ((f9219.ge.'A').and.(f9219.le.'J')) then
        hand = 3             ! mixed container
      else
        hand = 4             ! not handling mail
      end if

C     Assign weight increment
      if (hand.eq.1) then
        if (actv.ge.1000) then
          iw = weight(f165,if166,if167,ct_nowgt,nw)  !  Subroutine assigns weight increment
        else
          iw = nw  !  Special service activities assumed to have no record weight
        end if
      else
        iw = nw
      end if

C     Sum direct tally dollar weights by weight increment, cost pool, and F9806 activity code
      if (((if9806.ge.10).and.(if9806.le.4950)).or.((if9806.ge.5300).and.(if9806.le.5480))) then
        if (iact2.gt.0) then
          dir9806(iw,modgrp,iact2) = dir9806(iw,modgrp,iact2) + dlrs ! direct
        else
          print *,'iact2 = 0; f9806 = ',if9806
        end if
      end if

C     Single piece being handled, Assign to A matrix
      if ((hand.eq.1).and.((iitem.eq.0).and.(icon.eq.0))) then
        if (iact.gt.0) then
          if ((modgrp.gt.0).and.(modgrp.le.nmod)) then
            adols(iw,modgrp,iact,ishp)=adols(iw,modgrp,iact,ishp) + dlrs
            atot = atot + dlrs
            acnt = acnt + 1
          else
            print *,' bad MODS in matrix A ',f114, modgrp, dlrs
          end if
        else                 !  Not handling mail
          print *,' bad activity in matrix A ',actv
          if (modgrp.gt.0) then
            jdols(modgrp) = jdols(modgrp) + dlrs
            jtot = jtot + dlrs
            jcnt = jcnt + 1
          end if
        end if

C*************************************************************************************
C     Not-handling mail tallies -- assign to J matrix

      else if (hand.eq.4) then

        jdols(modgrp) = jdols(modgrp) + dlrs
        jtot = jtot + dlrs
        jcnt = jcnt + 1

C*************************************************************************************
C     Item being handled: separate items with direct activity codes from others

      else if ((f9214.ge.'A').and.(f9214.le.'P')) then
        if (hand.eq.1) then
          imat = 1           ! "B" matrix - identical, top piece, or counted item
        else if (hand.eq.2) then
          imat = 3           ! "D" matrix - mixed, empty item
        else
          print *,'problem item in modgrp = ',modgrp
          imat = 0
        end if
```

```
      "D" matrix: mixed or empty item
            if (imat.eq.3) then
               ddols(modgrp,iitem) = ddols(modgrp,iitem) + dlrs
               dtot = dtot + dlrs
               dcnt = dcnt + 1

      "B" matrix: identical or top piece rule (direct item)
            else if (imat.eq.1) then
               bdols(iw,modgrp,iitem,iact) =
     &            bdols(iw,modgrp,iitem,iact) + dlrs
               btot = btot + dlrs
               bcnt = bcnt + 1
            else                  !  Not handling mail
               print *,' imat 0 in modgrp = ',actv
               jdols(modgrp) = jdols(modgrp) + dlrs
               jtot = jtot + dlrs
               jcnt = jcnt + 1
            end if
c*****************************************End Item*****************************
c     Container being handled: separate containers with direct activity codes from others
         else if (icon.gt.0) then

            if (modgrp.gt.0) then

               flag2=.false.

               if (f9901(1:1).eq.'%') then
                  read(rec(340:406),451,iostat=ier) counts
               else
                  read(rec(339:406),450,iostat=ier) counts
               end if
450            format(5(1x,f3.0),16f3.0)
451            format(f3.0,4(1x,f3.0),16f3.0)


               if (ier.ne.0) then
                  flag2 = .true.
                  j = 340
                  do i = 1, ncsi
                     counts(i) = 0.
                  end do
                  ier = 0
               end if

               sum = 0.
               do i = 1, ncsi
                  sum = sum + counts(i)
               end do

c     "F" matrix: identical mail in container (direct container)
               if (hand.eq.1) then
                  fdols(iw,modgrp,icon,iact) =
     &               fdols(iw,modgrp,icon,iact) + dlrs
                  ftot = ftot + dlrs
                  fcnt = fcnt + 1

c     "H" matrix:  Uncounted, empty, or contents read error
               else if ((sum.eq.0.).or.flag2) then
                  hdols(modgrp,icon) = hdols(modgrp,icon) + dlrs
                  htot = htot + dlrs
                  hcnt = hcnt + 1

c     "G" matrix:  container contents are "identified"
               else if (sum.gt.0.) then
                  do icsi = 1, ncsi
                     gdols(modgrp,icon,icsi) = gdols(modgrp,icon,icsi) +
     &                  (counts(icsi)/sum) * dlrs
                  end do
                  gtot = gtot + dlrs
                  gcnt = gcnt + 1
               end if
            end if

C*************************************End Container*****************************
c     Any remaining tallies considered not handling mail
         else
            jdols(modgrp) = jdols(modgrp) + dlrs
            jtot = jtot + dlrs
```

```
         jcnt = jcnt + 1
         print *,'bad hand/mat in modgrp = ',modgrp

      end if

   end do
   print *,' read exit = ',ier,' with ',cnt,' records '

************************************End Read Loop*********************************
   Generate a distribution key to distribute no weight tallies over all direct pieces, identical/top piece
   items, and identical/top piece containers - Only used when all other distribution attempts fail
   Direct pieces
   do imod = 1, nmod
      do iact = 1, nact2
         do iw = 1, nw2
            do ishp = 1, nshp
               nowgt_key(iw,imod,iact) = nowgt_key(iw,imod,iact) + adols(iw,imod,iact,ishp)
            end do
         end do
      end do
   end do
   Identical/top piece items
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw2
            do iitem = 1, nitem
               nowgt_key(iw,imod,iact) = nowgt_key(iw,imod,iact) + bdols(iw,imod,iitem,iact)
            end do
         end do
      end do
   end do
   Identical/top piece containers
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw2
            do icon = 1, ncon
               nowgt_key(iw,imod,iact) = nowgt_key(iw,imod,iact) + fdols(iw,imod,icon,iact)
            end do
         end do
      end do
   end do

   Redistribute no weight direct single piece costs
   do iact = begmail, nact2
      do imod = 1, nmod
         do ishp = 1, nshp
            if (adols(nw,imod,iact,ishp).gt.0.0) then
               sum = 0.0
               do iw = 1, nw2 !  Distribute over all weight increments
                  sum = sum + adols(iw,imod,iact,ishp)
               end do
               if (sum.gt.0) then
                  do iw = 1, nw2
                     adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                  adols(nw,imod,iact,ishp)*adols(iw,imod,iact,ishp)/sum
                  end do
                  adols(nw,imod,iact,ishp) = 0.0
               end if
            end if
         end do
      end do
   end do

   Residual distribution of direct single piece no weight costs
   do iact = begmail, nact2
      do imod = 1, nmod
         do ishp = 1, nshp
            if (adols(nw,imod,iact,ishp).gt.0.0) then
               sum = 0.0
               do iw = 1, nw2
                  actwgt(iw) = 0.0
               end do
               do j = 1, nmod !  Distbribute over all cost pools
                  do iw = 1, nw2 !  Distribute over all weight increments
                     actwgt(iw) = actwgt(iw) + adols(iw,j,iact,ishp)
                     sum = sum + adols(iw,j,iact,ishp)
                  end do
               end do
               if (sum.gt.0) then
                  do iw = 1, nw2
```

```
            adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &         adols(nw,imod,iact,ishp)*actwgt(iw)/sum
          end do
          adols(nw,imod,iact,ishp) = 0.0
       else
          if (adols(nw,imod,iact,ishp).gt.0.) then
             print*,'Level 3 distribution of act = ',acodes(iact)
             do k = begmail, nact2
                do iw = 1, nw2
                   actshr2(iw,k) = 0.
                end do
             end do
             do k = 1, nact2 !  Distribute over all activity codes within same subclass
                if (class(k).eq.class(iact)) then !  Same subclass
                   do iw = 1, nw2 !  Distribute over all weight increments
                      actshr2(iw,k) = actshr2(iw,k) + adols(iw,imod,k,ishp)
                      sum = sum + adols(iw,imod,k,ishp)
                   end do
                end if
             end do
             if (sum.gt.0.) then
                do k = begmail, nact2
                   do iw = 1, nw2
                      adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                   adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                   end do
                end do
                adols(nw,imod,iact,ishp) = 0.0
             else
                print*,'Level 4 distribution of act = ',acodes(iact)
                do k = begmail, nact2
                   do iw = 1, nw2
                      actshr2(iw,k) = 0.
                   end do
                end do
                do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                   if (class(k).eq.class(iact)) then !  Same subclass
                      do j = 1,nmod !  Distribute over all cost pools
                         do iw = 1, nw2 !  Disbribute over all weight increments
                            actshr2(iw,k) = actshr2(iw,k) + adols(iw,j,k,ishp)
                            sum = sum + adols(iw,j,k,ishp)
                         end do
                      end do
                   end if
                end do
                if (sum.gt.0.) then
                   do k = begmail, nact2
                      do iw = 1, nw2
                         adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                      adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                      end do
                   end do
                   adols(nw,imod,iact,ishp) = 0.0
                else
                   if (ishp.eq.1) then !  assign card directely to < 1/2 oz increment
                      print*, 'Assign card directly to < 1/2 oz increment' !
                      adist(1,imod,iact,ishp) = adist(1,imod,iact,ishp) +
     &                   adols(nw,imod,iact,ishp)
                      adols(nw,imod,iact,ishp) = 0.0
                   else
                      print*,'Level 5 distribution of act = ',acodes(iact)
                      do k = begmail, nact2
                         do iw = 1, nw2
                            actshr2(iw,k) = 0.
                         end do
                      end do
                      do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                         if (class_code(k).eq.class_code(iact)) then !  Same subclass
                            do j = 1,nmod !  Distribute over all cost pools
                               do iw = 1, nw2 !  Disbribute over all weight increments
                                  actshr2(iw,k) = actshr2(iw,k) + adols(iw,j,k,ishp)
                                  sum = sum + adols(iw,j,k,ishp)
                               end do
                            end do
                         end if
                      end do
                      if (sum.gt.0.) then
                         do k = begmail, nact2
                            do iw = 1, nw2
                               adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
```

```
&                                        adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                                    end do
                                 end do
                                 adols(nw,imod,iact,ishp) = 0.0
                              else
                                 print*,'unable to distribute no weight for ',
                                    imod,' act = ',acodes(iact), ' cost = ', adols(nw,imod,iact,ishp)
                              end if
                           end if
                        end if
                     end if
                  end if
               end if
            end do
         end do
      end do

      Add in redistributed no weight direct single piece costs
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               do ishp = 1, nshp
                  adols(iw,imod,iact,ishp) = adols(iw,imod,iact,ishp) + adist(iw,imod,iact,ishp)
               end do
            end do
         end do
      end do

      Redistribute no weight identical/top piece item costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do iitem = 1, nitem
               if (bdols(nw,imod,iitem,iact).gt.0) then
                  sum = 0.0
                  do iw = 1, nw2 !  Distribute over all weight increments
                     sum = sum + bdols(iw,imod,iitem,iact)
                  end do
                  if (sum.gt.0.0) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                          bdols(nw,imod,iitem,iact)*bdols(iw,imod,iitem,iact)/sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  end if
               end if
            end do
         end do
      end do

c     Residual distribution of identical/top piece items no weight costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do iitem = 1, nitem
               if (bdols(nw,imod,iitem,iact).gt.0.0) then
                  sum = 0.0
                  do iw = 1, nw2
                     actwgt(iw) = 0.0
                  end do
                  do j = 1, nitem !  Distribute over all item types
                     do iw = 1, nw2 !  Distribute over all weight increments
                        actwgt(iw) = actwgt(iw) + bdols(iw,imod,j,iact)
                        sum = sum + bdols(iw,imod,j,iact)
                     end do
                  end do
                  if (sum.gt.0) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                          bdols(nw,imod,iitem,iact)*actwgt(iw)/sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  else
                     if (bdols(nw,imod,iitem,iact).gt.0.0) then
                        print*, 'Level 3 b distribution of act = ', acodes(iact)
                        do iw = 1, nw2
                           actwgt(iw) = 0.
                        end do
                        do k = 1, nmod !  Distribute over all cost pools
                           do j = 1,nitem !  Distribute over all item types
                              do iw = 1, nw2 !  Distribute over all weight increments
```

```
                        actwgt(iw) = actwgt(iw) + bdols(iw,k,j,iact)
                        sum = sum + bdols(iw,k,j,iact)
                     end do
                  end do
               end do
               if (sum.gt.0.) then
                  do iw = 1, nw2
                     bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                       bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                  end do
                  bdols(nw,imod,iitem,iact) = 0.0
               else
                  print*, 'Level 4 b distribution of act = ', acodes(iact)
                  do iw = 1, nw2
                     actwgt(iw) = 0.
                  end do
                  do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                     if (class(k).eq.class(iact)) then !  Same subclass
                        do j = 1, nmod !  Distribute over all cost pools
                           do l = 1,nitem !  Distribute over all item types
                              do iw = 1, nw2 !  Distribute over all weight increments
                                 actwgt(iw) = actwgt(iw) + bdols(iw,j,l,k)
                                 sum = sum + bdols(iw,j,l,k)
                              end do
                           end do
                        end do
                     end if
                  end do
                  if (sum.gt.0.) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                          bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  else
                     print*, 'Level 5 b distribution of act = ', acodes(iact)
                     do iw = 1, nw2
                        actwgt(iw) = 0.
                     end do
                     do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                        if (class_code(k).eq.class_code(iact)) then !  Same subclass
                           do j = 1, nmod !  Distribute over all cost pools
                              do l = 1,nitem !  Distribute over all item types
                                 do iw = 1, nw2 !  Disbribute over all weight increments
                                    actwgt(iw) = actwgt(iw) + bdols(iw,j,l,k)
                                    sum = sum + bdols(iw,j,l,k)
                                 end do
                              end do
                           end do
                        end if
                     end do
                     if (sum.gt.0.) then
                        do iw = 1, nw2
                           bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                             bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                        end do
                        bdols(nw,imod,iitem,iact) = 0.0
                     else
                        print*,'unable to distribute no weight for b, ',
&                          imod,' act = ',acodes(iact), ' cost = ', bdols(nw,imod,iitem,iact)
                     end if
                  end if
               end if
            end if
         end if
      end do
   end do
end do

c  Final no weight redistribution for identical/top piece items - using no wgt key
   do iact = begmail, nact2
      do imod = 1, nmod
         do iitem = 1, nitem
            if (bdols(nw,imod,iitem,iact).gt.0) then
               print*, 'b dols nowgt key dist for ', acodes(iact)
               sum = 0.0
               do iw = 1, nw2 !  Disbribute over all weight increments
                  sum = sum + nowgt_key(iw,imod,iact)
               end do
```

```
      if (sum.gt.0.0) then
         do iw = 1, nw2
            bdist(iw,imod,iitem,iact) = bdist(iw,iitem,icon,iact) +
     &         bdols(nw,imod,iitem,iact)*nowgt_key(iw,imod,iact)/sum
         end do
         bdols(nw,imod,iitem,iact) = 0.0
      else
         print*, 'Level 2 distrib for b, nowgt key ', acodes(iact)
         sum = 0.0
         do iw = 1, nw2
            actwgt(iw) = 0.0
         end do
         do j = 1, nmod !  Distribute over all cost pools
            do iw = 1, nw2 !  Disbribute over all weight increments
               actwgt(iw) = actwgt(iw) + nowgt_key(iw,j,iact)
               sum = sum + nowgt_key(iw,j,iact)
            end do
         end do
         if (sum.gt.0) then
            do iw = 1, nw2
               bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &            bdols(nw,imod,iitem,iact)*actwgt(iw)/sum
            end do
            bdols(nw,imod,iitem,iact) = 0.0
         else
            if (bdols(nw,imod,iitem,iact).gt.0.) then
               print*,'Level 3 nowgt_key distribution of b act = ',acodes(iact)
               do k = begmail, nact2
                  do iw = 1, nw2
                     actshr2(iw,k) = 0.
                  end do
               end do
               do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                  if (class(k).eq.class(iact)) then !  Same subclass
                     do j = 1,nmod !  Distribute over all cost pools
                        do iw = 1, nw2 !  Disbribute over all weight increments
                           actshr2(iw,k) = actshr2(iw,k) + nowgt_key(iw,j,k)
                           sum = sum + nowgt_key(iw,j,k)
                        end do
                     end do
                  end if
               end do
               if (sum.gt.0.) then
                  do k = begmail, nact2
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                     bdols(nw,imod,iitem,iact) * actshr2(iw,k) / sum
                     end do
                  end do
                  bdols(nw,imod,iitem,iact) = 0.0
               else
                  print*, 'Level 5 b distribution of act = ', acodes(iact)
                  do iw = 1, nw2
                     actwgt(iw) = 0.
                  end do
                  do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                     if (class_code(k).eq.class_code(iact)) then !  Same subclass
                        do j = 1, nmod !  Distribute over all cost pools
                           do iw = 1, nw2 !  Disbribute over all weight increments
                              actwgt(iw) = actwgt(iw) + nowgt_key(iw,j,k)
                              sum = sum + nowgt_key(iw,j,k)
                           end do
                        end do
                     end if
                  end do
                  if (sum.gt.0.) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                     bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  else
                     print*,'unable to distribute no weight for b, ',
     &                  imod,' act = ',acodes(iact), ' cost = ', bdols(nw,imod,iitem,iact)
                  end if
               end if
            end if
         end if
      end if
   end if
end if
```

```
      end do
    end do
  end do

  Add in redistributed no weight identical/top piece item costs
  do iact = 1, nact2
    do imod = 1, nmod
      do iw = 1, nw
        do iitem = 1, nitem
          bdols(iw,imod,iitem,iact) = bdols(iw,imod,iitem,iact) + bdist(iw,imod,iitem,iact)
          bdist(iw,imod,iitem,iact) = 0.0
        end do
      end do
    end do
  end do

  Residual distribution of identical/top piece container no weight costs
  do iact = begmail, nact2
    do imod = 1, nmod
      do icon = 1, ncon
        if (fdols(nw,imod,icon,iact).gt.0) then
          sum = 0.0
          do iw = 1, nw2
            sum = sum + fdols(iw,imod,icon,iact)
          end do
          if (sum.gt.0.0) then
            do iw = 1, nw2 !  Distribute over all weight increments
              fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&               fdols(nw,imod,icon,iact)*fdols(iw,imod,icon,iact)/sum
            end do
            fdols(nw,imod,icon,iact) = 0.0
          end if
        end if
      end do
    end do
  end do

  Residual distribution of identical/top piece container no weight costs
  do iact = begmail, nact2
    do imod = 1, nmod
      do icon = 1, ncon
        if (fdols(nw,imod,icon,iact).gt.0.0) then
          sum = 0.0
          check = 0.0
          do iw = 1, nw2
            actwgt(iw) = 0.0
          end do
          do j = 1, nmod !  Distbribute over all cost pools
            do iw = 1, nw2 !  Distribute over all weight increments
              actwgt(iw) = actwgt(iw) + fdols(iw,j,icon,iact)
              sum = sum + fdols(iw,j,icon,iact)
            end do
          end do
          if (sum.gt.0) then
            do iw = 1, nw2
              fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&                 fdols(nw,imod,icon,iact)*actwgt(iw)/sum
            end do
            fdols(nw,imod,icon,iact) = 0.0
          else
            if (fdols(nw,imod,icon,iact).gt.0.) then
              print*,'Level 3 distribution of f act = ',acodes(iact)
              do k = begmail, nact2
                do iw = 1, nw2
                  actshr2(iw,k) = 0.
                end do
              end do
              do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                if (class(k).eq.class(iact)) then !  Same subclass
                  do iw = 1, nw2 !  Distribute over all weight increments
                    actshr2(iw,k) = actshr2(iw,k) + fdols(iw,imod,icon,k)
                    sum = sum + fdols(iw,imod,icon,k)
                  end do
                end if
              end do
              if (sum.gt.0.) then
                do k = begmail, nact2
                  do iw = 1, nw2
                    fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&                     fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
```

```
                  end do
               end do
               fdols(nw,imod,icon,iact) = 0.0
            else
               print*,'Level 4 distribution f of act = ',acodes(iact)
               do k = begmail, nact2
                  do iw = 1, nw2
                     actshr2(iw,k) = 0.
                  end do
               end do
               do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                  if (class(k).eq.class(iact)) then !  Same subclass
                     do j = 1,nmod !  Distribute over all cost pools
                        do iw = 1, nw2 !  Distribute over all weight increments
                           actshr2(iw,k) = actshr2(iw,k) + fdols(iw,j,icon,k)
                           sum = sum + fdols(iw,j,icon,k)
                        end do
                     end do
                  end if
               end do
               if (sum.gt.0.) then
                  do k = begmail, nact2
                     do iw = 1, nw2
                        fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&                          fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                     end do
                  end do
                  fdols(nw,imod,icon,iact) = 0.0
               else
                  print*,'Level 5 distribution f of act = ',acodes(iact)
                  do k = begmail, nact2
                     do iw = 1, nw2
                        actshr2(iw,k) = 0.
                     end do
                  end do
                  do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                     if (class_code(k).eq.class_code(iact)) then !  Same subclass
                        do j = 1,nmod !  Distribute over all cost pools
                           do iw = 1, nw2 !  Distribute over all weight increments
                              actshr2(iw,k) = actshr2(iw,k) + fdols(iw,j,icon,k)
                              sum = sum + fdols(iw,j,icon,k)
                           end do
                        end do
                     end if
                  end do
                  if (sum.gt.0.) then
                     do k = begmail, nact2
                        do iw = 1, nw2
                           fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&                             fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                        end do
                     end do
                     fdols(nw,imod,icon,iact) = 0.0
                  end if
               end if
            end if
         end if
      end if
   end do
end do

c  Final no weight redistribution for identical/top piece containers - using no wgt key
do iact = begmail, nact2
   do imod = 1, nmod
      do icon = 1, ncon
         if (fdols(nw,imod,icon,iact).gt.0) then
            print*, 'f dols nowgt key dist for ', acodes(iact)
            sum = 0.0
            do iw = 1, nw2 !  Distribute over all weight increments
               sum = sum + nowgt_key(iw,imod,iact)
            end do
            if (sum.gt.0.0) then
               do iw = 1, nw2
                  fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&                    fdols(nw,imod,icon,iact)*nowgt_key(iw,imod,iact)/sum
               end do
               fdols(nw,imod,icon,iact) = 0.0
            else
```

```
                print*, 'Level 2 distrib for f, nowgt key ', acodes(iact)
                sum = 0.0
                do iw = 1, nw2
                   actwgt(iw) = 0.0
                end do
                do j = 1, nmod !  Distribute over all cost pools
                   do iw = 1, nw2 !  Distribute over all weight increments
                      actwgt(iw) = actwgt(iw) + nowgt_key(iw,j,iact)
                      sum = sum + nowgt_key(iw,j,iact)
                   end do
                end do
                if (sum.gt.0) then
                   do iw = 1, nw2
                      fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
&                        fdols(nw,imod,icon,iact)*actwgt(iw)/sum
                   end do
                   fdols(nw,imod,icon,iact) = 0.0
                else
                   if (fdols(nw,imod,icon,iact).gt.0.) then
                      print*,'Level 3 nowgt_key distribution of f act = ',acodes(iact)
                      do k = begmail, nact2
                         do iw = 1, nw2
                            actshr2(iw,k) = 0.
                         end do
                      end do
                      do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                         if (class(k).eq.class(iact)) then !  Same subclass
                            do j = 1,nmod !  Distribute over all cost pools
                               do iw = 1, nw2 !  Distribute over all weight increments
                                  actshr2(iw,k) = actshr2(iw,k) + nowgt_key(iw,j,k)
                                  sum = sum + nowgt_key(iw,j,k)
                               end do
                            end do
                         end if
                      end do
                      if (sum.gt.0.) then
                         do k = begmail, nact2
                            do iw = 1, nw2
                               fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
                                  fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                            end do
                         end do
                         fdols(nw,imod,icon,iact) = 0.0
                      else
                         print*,'unable to distribute no weight f for ',
&                           imod,' act = ',acodes(iact), ' cost = ', fdols(nw,imod,icon,iact)
                      end if
                   end if
                end if
             end if
          end if
       end do
    end do
 end do

 Add in redistributed no weight identical/top piece container costs
 do iact = 1, nact2
    do imod = 1, nmod
       do iw = 1, nw
          do icon = 1, ncon
             fdols(iw,imod,icon,iact) = fdols(iw,imod,icon,iact) + fdist(iw,imod,icon,iact)
             fdist(iw,imod,icon,iact) = 0.0
          end do
       end do
    end do
 end do

 Distribute mixed/empty item costs ("D" matrix) using direct item costs ("B" matrix) as a distribution key
 over all activity codes and weight increments within cost pool and item type

 print *,' distributing D '
 do imod = begmod, nmod
    if (imod.ne.1) then    !  Excludes Platform
       do iitem = 1, nitem
          if (iitem.ne.9) then !  Exclude green sacks
             if (ddols(imod,iitem).gt.0.) then
                sum = 0.
                do iact = 1, nact2 !  Distribute over all activity codes
                   do iw = 1, nw !  Distribute over all weight increments
                      sum = sum + bdols(iw,imod,iitem,iact)
```

```
                  end do
               end do
               if (sum.gt.0) then
                  do iact = 1, nact2
                     do iw = 1, nw
                        cdist(iw,imod,iitem,iact) = cdist(iw,imod,iitem,iact) +
                           ddols(imod,iitem) * bdols(iw,imod,iitem,iact) / sum
                     end do
                  end do
                  ddols(imod,iitem) = 0.
               end if
            end if
         end if
      end do
   end if
end do

Distribute remaining mixed/empty item costs ("D" matrix) over all activity codes, DBMC categories,
and cost pools within item type using direct item costs ("B" matrix)

do iitem = 1, nitem
   if (iitem.ne.9) then   ! Exclude green sacks
      do imod = begmod, nmod
         if (ddols(imod,iitem).gt.0.) then
            sum = 0
            do iact = 1, nact2
               do iw = 1, nw
                  actshr(iw,iact) = 0.
               end do
            end do
            do iact = 1, nact2 ! Distribute over all activity codes
               do j = begmod, nmod ! Distribute over all cost pools
                  do iw = 1, nw ! Distribute over all weight increments
                     actshr(iw,iact) = actshr(iw,iact) + bdols(iw,j,iitem,iact)
                     sum = sum + bdols(iw,j,iitem,iact)
                  end do
               end do
            end do
            if (sum.gt.0.) then
               do iact = 1, nact2
                  do iw = 1, nw
                     cdist(iw,imod,iitem,iact) = cdist(iw,imod,iitem,iact) +
&                       ddols(imod,iitem) * actshr(iw,iact) / sum
                  end do
               end do
               ddols(imod,iitem) = 0.
            else
               print *,'L2 unable to dist D dols for iitem = ',iitem,', ',ddols(imod,iitem)
            end if
         end if
      end do
   end if
end do

Sum distributed mixed/empty item costs in "D" distribution matrix
do iact = 1, nact2
   do iitem = 1, nitem
      do imod = begmod, nmod
         do iw = 1, nw
            ddist(iw,imod,iact) = ddist(iw,imod,iact) + cdist(iw,imod,iitem,iact)
         end do
      end do
   end do
end do


Distribute "identified" container costs ("G" matrix)

do imod = begmod, nmod    ! Initial distribution within cost pools

   if (imod.ne.1) then    ! Excludes Platform

      do icsi = 1, ncsi
         if (icsi.le.5) then ! Loose shapes distributed based upon direct piece costs ("A" matrix)
            sum = 0.
            distsum = 0.
            do iact = 1, nact2 ! Distribute over all activity codes
               do iw = 1, nw ! Distribute over all weight increments
                  sum = sum + adols(iw,imod,iact,icsi)
               end do
```

```
      end do
      if (sum.gt.0.) then
         do icon = 1, ncon
            if (gdols(imod,icon,icsi).gt.0.) then
               do iact = 1, nact2
                  do iw = 1, nw
                     gdist(iw,imod,icon,iact) =
     &                  gdist(iw,imod,icon,iact) +
     &                  gdols(imod,icon,icsi) *
     &                  adols(iw,imod,iact,icsi) / sum
                  end do
               end do
            end if
         end do
      else          ! distribute over all cost pools, activity codes, and weight increments
         do iact = 1, nact2 !  Distribute over all activity codes
            do i = begmod, nmod !  Distribute over all cost pools
               do iw = 1 , nw !  Distribute over all weight increments
                  distsum = distsum + adols(iw,i,iact,icsi)
               end do
            end do
         end do
         if (distsum.gt.0) then
            do iact = 1, nact2
               do icon = 1, ncon
                  do i = begmod, nmod
                     do iw = 1, nw
                        gdist(iw,imod,icon,iact) =
     &                     gdist(iw,imod,icon,iact) +
     &                     gdols(imod,icon,icsi) *
     &                     adols(iw,i,iact,icsi)/distsum
                     end do
                  end do
               end do
            end do
         else       !  Undistributed costs included with uncounted/empty containers ("H" matrix)
            print *,'shape distribution empty: mod = ',imod,
     &         ', shape = ',icsi
            do icon = 1, ncon
               hdols(imod,icon) = hdols(imod,icon) +
                  gdols(imod,icon,icsi)
            end do
         end if
      end if
   else             ! Items distributed upon direct item costs ("B" matrix)
      iitem = icsi - nshp2  !  Distribute over all item types
      print *,' G dist',imod,' ',iitem
      if (iitem.eq.9) then !  Exclude green sacks
         do icon = 1,ncon
            if (gdols(imod,icon,icsi).gt.0.) then
               print *,'mxd green sack in container, group =',imod,' $ =',
     &            gdols(imod,icon,icsi)
            end if
         end do
      else
         sum = 0.
         distsum = 0.
         do iact = 1, nact2 !  Distribute over all activity codes
            do iw = 1, nw !  Distribute over all weight increments
               sum = sum + bdols(iw,imod,iitem,iact)
            end do
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do iact = 1, nact2
                     do iw = 1, nw
                        gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
     &                     gdols(imod,icon,icsi) * bdols(iw,imod,iitem,iact) / sum
                        ddist(iw,imod,iact) = ddist(iw,imod,iact) +
     &                     gdols(imod,icon,icsi) * bdols(iw,imod,iitem,iact) / sum
                     end do
                  end do
               end if
            end do
         else       ! distribute over all cost pools, activity codes, and weight increment
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = begmod, nmod !  Distribute over all cost pools
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + bdols(iw,i,iitem,iact)
```

```
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
                     do i = begmod, nmod
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) =
&                             gdist(iw,imod,icon,iact) +
&                             gdols(imod,icon,icsi) *
&                             bdols(iw,i,iitem,iact)/distsum
                           ddist(iw,imod,iact) =
&                             ddist(iw,imod,iact) +
&                             gdols(imod,icon,icsi) *
&                             bdols(iw,i,iitem,iact)/distsum
                        end do
                     end do
                  end do
               end do
            else    !  Undistributed costs included with uncounted/empty containers ("H" matrix)
               print *,'shape distribution empty: mod = ',imod,
&                 ', shape = ',icsi
               do icon = 1, ncon
                  hdols(imod,icon) = hdols(imod,icon) +
&                    gdols(imod,icon,icsi)
               end do
            end if
         end if
      end if
   end if

end do

else if (imod.eq.1) then ! Distribute Platform over all allied labor cost pools, activity codes, and weight increments

   do icsi = 1, ncsi
      if (icsi.le.5) then ! Loose shapes distributed based upon direct piece costs ("A" matrix)
         sum = 0.
         do i = begmod, nmod !  Distribute over all cost pools
            do iact = 1, nact2 !  Distribute over all activity codes
               do iw = 1, nw !  Distribute over all weight increments
                  sum = sum + adols(iw,i,iact,icsi)
               end do
            end do
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do i = begmod, nmod
                     do iact = 1, nact2
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
&                             gdols(imod,icon,icsi) * adols(iw,i,iact,icsi) / sum
                        end do
                     end do
                  end do
               end if
            end do
         else          ! Platform items distributed upon identical/top piece item costs ("B" matrix)
            print *,'platform level 2 shape = ',icsi
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = begmod, nmod !  Distribute over all cost pools
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + adols(iw,i,iact,icsi)
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
                     do i = begmod, nmod
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) =
&                             gdist(iw,imod,icon,iact) +
&                             gdols(imod,icon,icsi) *
                              adols(iw,i,iact,icsi)/distsum
                        end do
                     end do
                  end do
```

```
            end do
         else       !  Undistributed Platform costs included with uncounted/empty containers ("H" matrix)
            print *,'shape distribution empty: mod = ',imod,
     &         ', shape = ',icsi
            do icon = 1, ncon
               hdols(imod,icon) = hdols(imod,icon) +
                  gdols(imod,icon,icsi)
            end do
         end if
      end if
   else             ! Platform items distributed upon identical/top piece item costs ("B" matrix)
      iitem = icsi - nshp2
      sum = 0.
      if ((iitem.eq.2).or.(iitem.eq.9)) then !  Exclude con-con and green sacks
         do icon = 1,ncon
            if (gdols(imod,icon,icsi).gt.0.) then
               print *,'mxd item=',iitem,' in container, group =',imod,' $ =',
     &            gdols(imod,icon,icsi)
                print *,'concon in platform container'
            end if
         end do
      else
         do i = begmod, nmod !  Distribute over all cost pools
            do iact = 1, nact2 !  Distribute over all activity codes
               do iw = 1, nw !  Distribute over all weight increments
                  sum = sum + bdols(iw,i,iitem,iact)
               end do
            end do
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do i = begmod, nmod
                     do iact = 1, nact2
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
     &                        gdols(imod,icon,icsi) * bdols(iw,i,iitem,iact) / sum
                           ddist(iw,imod,iact) = ddist(iw,imod,iact) +
     &                        gdols(imod,icon,icsi) * bdols(iw,i,iitem,iact) / sum
                        end do
                     end do
                  end do
               end if
            end do
         else
            print *,'platform level 2 item = ',iitem
            distsum = 0.
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = begmod, nmod !  Distribute over all cost pools
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + bdols(iw,i,iitem,iact)
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
                     do i = begmod, nmod
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) =
     &                        gdist(iw,imod,icon,iact) +
     &                        gdols(imod,icon,icsi) *
     &                        bdols(iw,i,iitem,iact)/distsum
                           ddist(iw,imod,iact) =
     &                        ddist(iw,imod,iact) +
     &                        gdols(imod,icon,icsi) *
     &                        bdols(iw,i,iitem,iact)/distsum
                        end do
                     end do
                  end do
               end do
            else    !  Undistributed Platform costs included with uncounted/empty containers ("H" matrix)
               print *,'item distribution empty: mod = ',imod,
     &            ', item = ',icsi
               do icon = 1, ncon
                  hdols(imod,icon) = hdols(imod,icon) +
     &               gdols(imod,icon,icsi)
               end do
            end if
         end if
```

```
            end if
          end if

       end do

    end if
 id do                    ! End of "identified" container ("G" matrix) distribution

Distribution adjustments
do iact = 1,nact2
   do iw = 1,nw
      do iitem = 1,nitem
         cdist(iw,1,iitem,iact) = cdist(iw,1,iitem,iact)*36326.0/(36326.0-75.1) !  Platform
         cdist(iw,2,iitem,iact) = cdist(iw,2,iitem,iact)*37640.0/(37640.0-393.4-49.5) !  Allied
      end do
      do icon = 1,ncon
         gdist(iw,1,icon,iact) = gdist(iw,1,icon,iact)*36326.0/(36326.0-75.1) !  Platform
         gdist(iw,2,icon,iact) = gdist(iw,2,icon,iact)*37640.0/(37640.0-393.4-49.5) !  Allied
      end do
   end do
end do

Sum up distribution factor for distribution of uncounted/empty costs ("H" matrix)
do iact = 1, nact2
   do icon = 1, ncon
      do imod = begmod, nmod
         do iw = 1, nw
            hkey(iw,imod,icon,iact) = fdols(iw,imod,icon,iact) + gdist(iw,imod,icon,iact)
         end do
      end do
   end do
end do

Distribute uncounted/empty containers ("H" matrix) using direct and distributed "identified"
container costs over all activity codes and weight increment within cost pool and
container type
do imod = begmod, nmod
   do icon = 1, ncon
      sum = 0.
      do iact = 1, nact2
         do iw = 1, nw
            sum = sum + hkey(iw,imod,icon,iact)
         end do
      end do
      if (sum.gt.0) then
         do iact = 1, nact2 !  Distribute over all activity codes
            do iw = 1, nw !  Distribute over all weight increments
               hdist(iw,imod,icon,iact) = hdist(iw,imod,icon,iact) +
&                 hdols(imod,icon) * hkey(iw,imod,icon,iact) / sum
            end do
         end do
         hdols(imod,icon) = 0.
      end if
   end do
end do

Distribute remaining uncounted/empty container costs ("H" matrix) over all activity codes, weight
increment, and cost pools within container type using direct/distributed "identified" container
costs

do icon = 1, ncon
   do imod = begmod, nmod
      if (hdols(imod,icon).gt.0.) then
         sum = 0.
         do iact = 1, nact2
            do iw = 1, nw
               actshr(iw,iact) = 0.
            end do
         end do
         do iact = 1, nact2 !  Distribute over all activity codes
            do j = begmod, nmod !  Distribute over all cost pools
               do iw = 1, nw !  Distribute over all weight increments
                  actshr(iw,iact) = actshr(iw,iact) + hkey(iw,j,icon,iact)
                  sum = sum + hkey(iw,j,icon,iact)
               end do
            end do
         end do
         if (sum.gt.0.) then
            do iact = 1, nact2
               do iw = 1, nw
```

```
                  hdist(iw,imod,icon,iact) = hdist(iw,imod,icon,iact) +
&                    actshr(iw,iact)/sum * hdols(imod,icon)
               end do
            end do
         else
            print *,' unable to dist h dols for imod = ',imod,
               ' icon = ',icon
         end if
      end if
   end do
end do

Sum up all costs (direct and redistributed) except not handling costs ("J" matrix)
Direct costs
do iact = 1, nact2
   do imod = begmod, nmod
      do iw = 1, nw
         result2(iw,imod,iact) = result2(iw,imod,iact) + dir9806(iw,imod,iact)
      end do
   end do
end do
Pieces
do ishp = 1, nshp
   do iact = 1, nact2
      do imod = begmod, nmod
         do iw = 1, nw
            result(iw,imod,iact) = result(iw,imod,iact) + adols(iw,imod,iact,ishp)
            resulta(iw,imod,iact) = resulta(iw,imod,iact) + adols(iw,imod,iact,ishp)
         end do
      end do
   end do
end do
Items
do iact = 1, nact2
   do iitem = 1, nitem
      do imod = begmod, nmod
         do iw = 1, nw
            result(iw,imod,iact) = result(iw,imod,iact) + bdols(iw,imod,iitem,iact)
               + cdols(iw,imod,iitem,iact)+cdist(iw,imod,iitem,iact)
            result2(iw,imod,iact) = result2(iw,imod,iact) + cdols(iw,imod,iitem,iact)
               +cdist(iw,imod,iitem,iact)
            resultb(iw,imod,iact) = resultb(iw,imod,iact) + bdols(iw,imod,iitem,iact)
&              + cdols(iw,imod,iitem,iact) +cdist(iw,imod,iitem,iact)
         end do
      end do
   end do
end do
Containers
do iact = 1, nact2
   do icon = 1, ncon
      do imod = begmod, nmod
         do iw = 1, nw
            result(iw,imod,iact) = result(iw,imod,iact) + fdols(iw,imod,icon,iact) +
&              gdist(iw,imod,icon,iact) + hdist(iw,imod,icon,iact)
            result2(iw,imod,iact) = result2(iw,imod,iact) + gdist(iw,imod,icon,iact)
&              + hdist(iw,imod,icon,iact)
            resultf(iw,imod,iact) = resultf(iw,imod,iact) + fdols(iw,imod,icon,iact) +
&              gdist(iw,imod,icon,iact)+ hdist(iw,imod,icon,iact)
         end do
      end do
   end do
end do

Adds break costs to not-handling
Calculated by taking dollar wgts by pool incl breaks (6521) - dollar wgts by pool without breaks

jdols(2) = jdols(2) + 273603.0 - 231878.0 !  Allied Oth
jdols(1) = jdols(1) + 217263.0 - 188655.0 !  Platform

Distribute not handling costs ("J" matrix) using all other costs ("results" matrix)
do imod = begmod, nmod
   if (imod.ge.3) then     !  Exclude allied and platform cost pools
      sum = 0.
      distsum = 0.
      do iact = 1, nact2   !  Distribute over all activity codes
         do iw = 1, nw     !  Distribute over all weight increments
            sum = sum + result(iw,imod,iact)
         end do
      end do
      if (sum.gt.0) then
```

```
         do iact = 1, nact2
            do iw = 1, nw
               work(iw,imod,iact) = work(iw,imod,iact) +
&                 jdols(imod) * result(iw,imod,iact) / sum
            end do
         end do
      else
         print *,' unable to distribute J dollars for ',imod
      end if
   else                    !  Distribute allied other and platform over all cost pools
      sum = 0.
      distsum = 0.
      do iact = 1, nact2   !  Distribute over all activity codes
         do iw = 1, nw     !  Distribute over all weight increments
            do i = begmod, nmod !  Distribute over all cost pools
               sum = sum + result(iw,i,iact)
            end do
         end do
      end do
      if (sum.gt.0) then
         do i = begmod, nmod
            do iact = 1, nact2
               do iw = 1, nw
                  work(iw,imod,iact) = work(iw,imod,iact) +
&                    jdols(imod) * result(iw,i,iact) / sum
               end do
            end do
         end do
      else
         print *,' unable to distribute J dollars for ',imod
      end if
   end if
end do

Sum distributed not handling costs ("J" matrix) into handling costs ("results" matrix)
do iact = 1, nact2
   do imod = begmod, nmod
      do iw = 1, nw
         result(iw,imod,iact) = result(iw,imod,iact) + work(iw,imod,iact)
         result2(iw,imod,iact) = result2(iw,imod,iact) + work(iw,imod,iact)
         resultj(iw,imod,iact) = work(iw,imod,iact)
         work(iw,imod,iact) = 0.
      end do
   end do
end do

Redistribute class-specific mixed mail costs over appropriate class-specific direct activity codes,
weight increment, and within cost pools
do imod = begmod,nmod
   do iact = 1,nmixcl
      do iw = 1, nw
         if (result(iw,imod,nact+iact).gt.0.0) then
            sum = 0.
            do i = 1,nact
               actshr3(i) = 0.
            end do
            do i = 1,nact !  Distribute over all direct activity codes
               do j = 1,nw !  Distribute over all weight increments
                  if (mixmap(i,iact).gt.0) then
                     sum = sum + result(j,imod,mixmap(i,iact))
                     actshr3(mixmap(i,iact)) = actshr3(mixmap(i,iact))
&                       + result(j,imod,mixmap(i,iact))
                  end if
               end do
            end do
            if (sum.gt.0.) then
               do i = 1,nact
                  if (mixmap(i,iact).gt.0) then
                     work(iw,imod,mixmap(i,iact)) =
&                       work(iw,imod,mixmap(i,iact)) +
&                       (result(iw,imod,nact+iact)*
&                       actshr3(mixmap(i,iact))/sum)
                  end if
               end do
               result(iw,imod,nact+iact) = 0.
            else
               sum = 0.
               do i = 1,nact
                  actshr3(i) = 0.
               end do
```

```
                  do i = 1,nact !  Distribute over all direct activity codes
                     do j = 1,nw !  Distribute over all weight increments
                        do k = begmod, nmod !  Distribute over all cost pools
                           if (mixmap(i,iact).gt.0) then
                              sum = sum + result(j,k,mixmap(i,iact))
                              actshr3(mixmap(i,iact)) = actshr3(mixmap(i,iact))
                                 + result(j,k,mixmap(i,iact))
                           end if
                        end do
                     end do
                  end do
                  if (sum.gt.0.) then
                     do i = 1,nact
                        if (mixmap(i,iact).gt.0) then
                           work(iw,imod,mixmap(i,iact)) =
     &                        work(iw,imod,mixmap(i,iact)) +
     &                        (result(iw,imod,nact+iact)*
     &                        actshr3(mixmap(i,iact))/sum)
                        end if
                     end do
                     result(iw,imod,nact+iact) = 0.
                  else
                     print*, 'Mix actv code not distributed ', acodes(nact+iact),
     &                    ' cost = ', result(iw,imod,nact+iact), ' pool ', modcodes(imod)
                  end if
               end if
            end if
         end do
      end do
   end do

c     Sum distributed class-specific mixed-mail costs into all other costs
      do iact = 1, nact
         do imod = begmod, nmod
            do iw = 1, nw
               result2(iw,imod,iact) = result2(iw,imod,iact) + work(iw,imod,iact)
               result(iw,imod,iact) = result(iw,imod,iact) + work(iw,imod,iact)
            end do
         end do
      end do

c     Compute volume-variable costs for all cost pools
      do imod = begmod, nmod
         sum = 0.
         do iact = 1, nact
            do iw = 1, nw
               sum = sum + result(iw,imod,iact)
            end do
         end do
         if (sum.gt.0.) then
            do iact = 1, nact
               do iw = 1, nw
                  varcost(iw,imod,iact) = varcost(iw,imod,iact) +
     &                 pooldols(imod)*variable(imod)*result(iw,imod,iact)/sum
                  novarcst(iw,imod,iact) = novarcst(iw,imod,iact) +
     &                 pooldols(imod)*variable(imod)*result2(iw,imod,iact)/sum
               end do
            end do
         else
            print *,'unable to distribute $ = ',pooldols(imod),
     &           ' for mods pool ',modcodes(imod)
         end if
      end do

c     Write out results to file
      open(80,file='bmc002by_wgt.data')
81    format(i3,i4,i3,8f18.9)

      do imod = begmod, nmod
         do iact = 1, nact
            do iw = 1, nw
               write (80,81) ldc1(imod), iact, iw, varcost(iw,imod,iact),
     &              novarcst(iw,imod,iact), result(iw,imod,iact),
                    resulta(iw,imod,iact), resultb(iw,imod,iact),
                    resultf(iw,imod,iact), resultj(iw,imod,iact),work(iw,imod,iact)
            end do
         end do
      end do

      Print *, ' Total Count and Dollars by Matrix '
```

```
   write (*,'(2x,a1,i6,f15.2)') 'A', acnt, atot
   write (*,'(2x,a1,i6,f15.2)') 'B', bcnt, btot
   write (*,'(2x,a1,i6,f15.2)') 'C', ccnt, ctot
   write (*,'(2x,a1,i6,f15.2)') 'D', dcnt, dtot
   write (*,'(2x,a1,i6,f15.2)') 'F', fcnt, ftot
   write (*,'(2x,a1,i6,f15.2)') 'G', gcnt, gtot
   write (*,'(2x,a1,i6,f15.2)') 'H', hcnt, htot
   write (*,'(2x,a1,i6,f15.2)') 'J', jcnt, jtot

   print *,'IOCS $ in  = ',dlrsin
   print *,'IOCS $ out = ',dlrsout
   print *,'5340 $ = ',dlrs5340in,' ',dlrs5340out

   end

------------------------------------------------------------------------------------------
   Assigns shape

   function shapeind(actv,f9635,f9805)

   integer*4 shapeind, actv
   character*1  f9635
   character*4 f9805

   if (((actv.ge.1000).and.(actv.lt.2000)).or.(actv.eq.5431).or.(actv.eq.5441)
  &     .or.(actv.eq.5451).or.(actv.eq.5461)) then
      if (((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K')) then
         shapeind = 1         ! cards
      else
         shapeind = 2         ! letters
      end if
   else if (((actv.ge.2000).and.(actv.lt.3000)).or.(actv.eq.5432).or.(actv.eq.5442)
  &     .or.(actv.eq.5452).or.(actv.eq.5462)) then
      shapeind = 3            ! flats
   else if (((actv.ge.3000).and.(actv.lt.4000)).or.(actv.eq.5433).or.(actv.eq.5443)
  &     .or.(actv.eq.5453).or.(actv.eq.5463)) then
      shapeind = 4            ! IPPs
   else if (((actv.ge.4000).and.(actv.lt.5000)).or.(actv.eq.5434).or.(actv.eq.5444)
  &     .or.(actv.eq.5454).or.(actv.eq.5464)) then
      shapeind = 5            ! parcels
   else
      shapeind = 6            ! other?
   end if

   if (actv.eq.5340) then
      shapeind = 6  !  other?
      if (((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K')) then
         shapeind = 1         ! cards
      end if
      if (f9635.eq.'A') then
         shapeind = 2  !  letters
      end if
      if ((f9635.eq.'D').or.(f9635.eq.'E')) then
         shapeind = 3  !  flats
      end if
      if ((f9635.eq.'F').or.(f9635.eq.'G').or.(f9635.eq.'J')) then
         shapeind = 4  !  IPPs
      end if
      if ((f9635.eq.'H').or.(f9635.eq.'I')) then
         shapeind = 5  !  parcels
      end if
   end if

   if ((actv.ge.10).and.(actv.lt.1000)) then
      if ((f9805(1:1).eq.'1').and.(((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K'))) then
         shapeind = 1  !  cards
      else if (f9805(1:1).eq.'1') then
         shapeind = 2  !  letters
      else if (f9805(1:1).eq.'2') then
         shapeind = 3  !  flats
      else if (f9805(1:1).eq.'3') then
         shapeind = 4  !  IPPs
      else if (f9805(1:1).eq.'4') then
         shapeind = 5  !  parcels
      else
         shapeind = 6  !  other?
      end if
   end if
```

```
   return
   end

------------------------------------------------------------------------------------------
   Assigns weight increment

    unction weight(f165,if166,if167,ct_nowgt,nw)

   character*1  f165
   integer*4    if166, if167, weight, ct_nowgt, nw

   weight = 0

   if (f165.eq.'A') then
      weight = 1              !  < 1/2 ounce
   else if (f165.eq.'B') then
      weight = 2              !  1 ounces
   else if (f165.eq.'C') then
      weight = 3              !  1 1/2 ounces
   else if (f165.eq.'D') then
      weight = 4              !  2 ounces
   else if (f165.eq.'E') then
      weight = 5              !  2 1/2 ounces
   else if (f165.eq.'F') then
      weight = 6              !  3 ounces
   else if (f165.eq.'G') then
      weight = 7              !  3 1/2 ounces
   else if (f165.eq.'H') then
      weight = 8              !  4 ounces
   else if (f165.eq.'I') then
      if (if166.eq.0) then    !  < 1 lb
         if (if167.gt.0) then
            weight = if167 + 4
         else
            weight = nw
            ct_nowgt = ct_nowgt + 1
         end if
      else if ((if166.eq.1).and.(if167.eq.0)) then
         weight = 20
      else if ((if166.gt.1).or.((if166.eq.1).and.(if167.gt.0))) then
         weight = 21
      else
         weight = nw
         ct_nowgt = ct_nowgt + 1
      end if
   else
      weight = nw
      ct_nowgt = ct_nowgt + 1
   end if

   return
   end
```

```
      program sumclass_bmc_wgt

c     Purpose:  Sum distributed volume-variable mail processing costs for BMCs to subclass
c               Costs are calculated in the Fortran program bmcproc00_wgt.f

      implicit none

      integer*4 nact, ncl, nmod, nshp, nmat, nshp2, nw

      parameter (nmod = 6)       !  Number of cost pools
      parameter (nact = 255)     !  Number of activity codes
      parameter (ncl = 60)       !  Number of subclasses
      parameter (nshp = 3)       !  Number of shapes
      parameter (nmat = 8)       !  Number of cost categories
      parameter (nshp2 = 5)      !  Number of shapes (class map)
      parameter (nw = 22)        !  Number of weight increments

      real*8    dollars(nmat,nw,nmod,nact)
      real*8   cdols(nmat,nmod,ncl,nshp,nw)

      integer*4 imod, iact, icl, i, j, k, shape, is
      integer*4 ier, shp(nact), iw
      integer*4 clmap(nact), mod(nmod), ldc1(nmod)

      character*14  grp(nmod)
      character*9 class(ncl), clcode, class2(ncl)
      character*4 temp
      character*4 acodes(nact), acin(nshp2)
      character*5 shapetype(nshp)/'1Ltr ','2Flt ','3Pcl '/

      ier = 0

c     Map of cost pools
      open(30,file='costpools.00.bmc.619')
32    format(i4,a14,i5)

      do i = 1, nmod
         read(30,32) mod(i), grp(i), ldc1(i)
      end do
      close(30)
      print *, 'BMC groups read'

c     Map of activity codes
      open(20,file='activity00.ecr.cra')
21    format(a4)

      do i = 1, nact
         read (20,21) acodes(i)
         is = shape(acodes(i))  !  Assign shape
         shp(i) = is
      end do
      print*, 'Read in activity codes '
      close(20)

c     Map of subclasses
      open(33,file='classes_intl.cra.new')
34    format(a9)
      do i = 1, ncl
         read(33,34) class(i)
         class2(i) = class(i)
      end do
      close(33)
      print*, 'Read in classes '

c     Maps activity codes to subclass
      open(35,file='classmap_intl.new')
36    format(a9,5(4x,a4))
      do i = 1, nact
         clmap(i) =  0
      end do
      do while (ier.eq.0)
         read(35,36,iostat=ier,end=101) clcode, acin
         do i = 1, nshp2
            j = 0
            if (acin(i).ne.'    ') then
               do iact = 1,nact
                  if (acodes(iact).eq.acin(i)) then
                     j = iact
                  end if
               end do
```

```
            if (j.gt.0) then
               temp = acin(i)
               if (((temp(2:2).eq.'6').or.(temp(2:2).eq.'7').or.
     &             (temp(2:2).eq.'8').or.(temp(1:2).eq.'54'))) then
                  clmap(j) = 17
               else
                  k = 0
                  do icl = 1,ncl
                     if (class2(icl).eq.clcode) then
                        k=icl
                     end if
                  end do
                  if (k.gt.0) then
                     clmap(j) = k
                  else
                     print *,' bad class code = ',clcode,' ',clcode
                  end if
               end if
            else
               print *,' activity code not found ',acin(i)
            end if
         end if
      end do
   end do
01 print *,' read exit of classmap = ',ier

  Initialize matrices
   do imod = 1, nmod
      do icl = 1, ncl
         do j = 1, nmat
            do is = 1, nshp
               do iw = 1, nw
                  cdols(j,imod,icl,is,iw) = 0.
               end do
            end do
         end do
      end do
   end do

      !ead in distributed cost data
      )pen(40,file='bmc002by_wgt.data')
1     format(10x,8f18.9)

      do imod = 1, nmod
         do iact = 1, nact
            do iw = 1, nw
               read (40,41) (dollars(j,iw,imod,iact),j=1,nmat)
            end do
         end do
      end do

C  Sum data to classes

      do j = 1, nmat
         do imod = 1, nmod
            do iact = 1, nact
               do iw = 1, nw
                  icl = clmap(iact) !  Subclass for corresponding activity code
                  is = shp(iact) !  Assign shape
                  if (icl.gt.0) then
                     cdols(j,imod,icl,is,iw) = cdols(j,imod,icl,is,iw)
    &                   + dollars(j,iw,imod,iact)
                  else
                     print *,' activity ',acodes(iact),' not in class map '
                  end if
               end do
            end do
         end do
      end do

C     Write out costs by subclass, cost pool, shape, and weight increment

      open(55,file='bmc00_wgt2.csv',recl=500)
5     format(a9,',',i2,',',a14,',',i2,',',a5,',',22(f18.9,','))

      do icl = 1, ncl
         do imod = 1, nmod
            do is = 1, nshp
               if ((icl.eq.1).or.(icl.eq.2).or.((icl.ge.8).and.(icl.le.11))) then
                  write(55,56) class(icl), imod, grp(imod), ldcl(imod), shapetype(is),
```

```
&               (cdols(1,imod,icl,is,iw), iw = 1, nw)
          end if
        end do
      end do
    end do

  .id
```

```
  Assign shape

  function shape(act)

  integer*4    shape
  character*4  act

  if (act(1:1).eq.'1') then
     shape = 1              ! Letters
  else if (act(1:1).eq.'2') then
     shape = 2              ! Plats
  else if ((act(1:1).eq.'3').or.(act(1:1).eq.'4')) then
     shape = 3              ! IPPs/Parcels
  else
     shape = 3              ! Other (Special Service)
     if (act.gt.'1000') then
        print*, 'No shape for actv ', act
     end if
  end if

  return
  end
```

```
program nmodproc00_wgt

Purpose:  Computes distributed volume-variable costs (USPS Method) for Non-MODS offices
          Adds additional dimension for various weight categories

implicit none

integer*4 nmod, nw, begmod, nw2
integer*4 nact, ishp, nshp, nmix, nmixcl, nact2
integer*4 nitem, nshp2, ncsi, ncon, begmail

parameter (nmod = 8)       !  Number of cost pools
parameter (begmod = 1)     !  Begining positionf for Non-MODS cost pools within map
parameter (nw =  22)       !  Number of weight increments (including no weight)
parameter (nw2 = 21)       !  Number of weight increments
parameter (nact = 255)     !  Number of direct activity codes
parameter (nshp = 6)       !  Number of shapes
parameter (nitem = 16)     !  Number of item types
parameter (nshp2 = 5)      !  Number of shapes (not including other)
parameter (ncon = 10)      !  Number of container types
parameter (nmix = 20)      !  Number of combined activity codes - for dist of counted items
parameter (ncsi = nshp2 + nitem) !  Number of "identified" container types (loose shapes + items)
parameter (begmail = 17)   ! Set this to the index of the first non-Spec Serv activity code
parameter (nmixcl = 20)    ! Number of class-specific mixed-mail codes
parameter (nact2 = 275)    ! Number of activity codes including class-specific mixed-mail

include 'iocs2000.h'

real*8     adols(nw,nmod,nact2,nshp) ! Handling direct single piece
real*8    adist(nw,nmod,nact2,nshp) ! Workspace for distribution of no weight single pieces
real*8     bdols(nw,nmod,nitem,nact2) ! Handling identical or top-piece item
real*8    bdist(nw,nmod,nitem,nact2) ! Workspace for distribution of no weight identical/top-piece items
real*8    cdist(nw,nmod,nitem,nact2) ! Workspace for distribution of matrix D
real*8     cdols(nw,nmod,nitem,nact2) ! Workspace for distributed costs from matrix D
real*8     ddols(nmod,nitem) ! Handling mixed/empty item
real*8     fdols(nw,nmod,ncon,nact2) ! Handling identical or top-piece container
real*8    fdist(nw,nmod,ncon,nact2) ! Workspace for distribution of no weight identical/top-piece containers
real*8     gdols(nmod,ncon,ncsi) ! Handling "identified" container
real*8    gdist(nw,nmod,ncon,nact2) ! G Matrix distributed to activity code
real*8     hdols(nmod,ncon) ! Handling uncounted/empty container
real*8    result(nw,nmod,nact2) ! Array to hold results
real*8    resulta(nw,nmod,nact2) ! Array to hold results for matrix A
real*8    resultb(nw,nmod,nact2) ! Array to hold results for matrix B, C, D
real*8    resultf(nw,nmod,nact2) ! Array to hold results for matrix F, G, H
real*8    resultj(nw,nmod,nact2) ! Array to hold distributed J matrix
real*8    work(nw,nmod,nact2) ! Array to hold distributed mixed class-specific
real*8    jdols(nmod)     ! Not Handling
real*8    counts(ncsi)
real*8    actshr(nw,nact2), actshr3(nact), actwgt(nw2), actshr2(nw,nact2)
real*8    dlrs, sum, distsum, rf9250, tot_dol, tot_dol2, check
real*8    bmix(nmod,nact2)
real*8    atot, btot, ctot, dtot, ftot, gtot, htot, jtot
real*8    pooldols(nmod)
real*8    variable(nmod)
real*8    varcost(nw,nmod,nact)
real*8    novarcst(nw,nmod,nact)
real*8    gfy,ovhfact,ovh6522,wgt,wgt6521,wgtall

logical flag

integer*4 acnt, bcnt, ccnt, dcnt, fcnt, gcnt, hcnt, jcnt
integer*4 ind, ldc, l
integer*4 cnt,npl,npnl,counted, class(nact2)
integer*4 i, j, imat, imod, icon, iact, icsi, iitem, shapeind, iw
integer*4 ier, k, class_code(nact2), poolcode(nmod), pool
integer*4 mapcodes(20)
integer*4 searchc, searchi, modgrp, hand, actv
integer*4 mixcodes(nmixcl)
integer*4 acodes(nact2), mixcount(nmixcl)
integer*4 mixmap(nact,nmixcl)
integer*4 ldcl(nmod)
integer*4 if166, if167, weight, ct_nowgt

character*14 modcodes(nmod)
character*1 codes(26)/'A','B','C','D','E','F','G','H','I','J','K',
&   'L','M','N','O','P','Q','R','S','T','U','V',
&   'W','X','Y','Z'/

logical flag2
```

```
      atot = 0.0
      btot = 0.0
      ctot = 0.0
      dtot = 0.0
      ftot = 0.0
      gtot = 0.0
      htot = 0.0
      jtot = 0.0
      acnt = 0
      bcnt = 0
      ccnt = 0
      dcnt = 0
      fcnt = 0
      gcnt = 0
      hcnt = 0
      jcnt = 0
      cnt = 0
      npl = 0
      npnl = 0
      counted = 0
      ier = 0
      gfy = 0.
      ovhfact = 0.
      ovh6522 = 0.
      wgt = 0.
      wgt6521 = 0.
      wgtall = 0.

      do i = 1, nmod
         pooldols(i) = 0.0
         variable(i) = 0.0
      end do

      do i = 1, 20
         mapcodes(i) = 0
      end do

      do i = 1, nmixcl
         mixcodes(i) = 0
         mixcount(i) = 0
      end do

C     Map of activity codes
      open(20,file='activity00.ecr.cra')
21    format(i4,i6,i5)
      do i=1,nact2
         read (20,21) acodes(i), class(i), class_code(i)
      end do
      print *,'read activity map'
      close(20)

C     Map of class specific mixed-mail activity codes
      open(20,file='mixclass.intl')
      do i = 1,nmixcl
         read (20,21) mixcodes(i)
      end do
      print *,'read mixed item code list'
      close(20)

      do i = 1,nact
         do j = 1,nmixcl
            mixmap(i,j) = 0
         end do
      end do

C     Maps class specific mixed-mail activity codes to appropriate direct activity codes
      open(20,file='mxmail.intl.dat')
23    format(20i4)

      do while (ier.eq.0)
         read (20,23,iostat=ier,end=75) mapcodes
         i = searchi(mixcodes,nmixcl,mapcodes(1))
         if (i.gt.0) then
            flag = .true.
            ind = 1
            do while ((flag).and.(ind.lt.20))
               ind = ind + 1
               if (mapcodes(ind).gt.0) then
                  j = searchi(acodes,nact,mapcodes(ind))
                  if (j.gt.0) then
```

```
                mixcount(i) = mixcount(i) + 1
                mixmap(mixcount(i),i) = j
             else
                print *,' Direct mail code did not map ',mapcodes(ind)
             end if
          else
             flag = .false.
          end if
       end do
    else
       print *,' Mixed mail code did not map ',mapcodes(1)
    end if
 end do
5 print *,' read mixed-mail map with exit code = ',ier
  close(20)

  Map of cost pool dollars and variabilities by cost pool
  open(20,file='costpools.00.nmod.619')
4 format(i4,a14,i5,f9.0,f7.2)
  do i = 1,nmod
     read(20,24) poolcode(i), modcodes(i), ldc1(i), pooldols(i), variable(i)
  end do
  close(20)

  Initialize matrices

  do iw = 1,nw
     do imod = 1,nmod
        do iact = 1,nact
           varcost(iw,imod,iact) = 0.
           novarcst(iw,imod,iact) = 0.
        end do
     end do
  end do
  do ishp = 1, nshp
     do iact = 1, nact2
        do imod = 1, nmod   ! a matrix
           do iw = 1, nw
              adols(iw,imod,iact,ishp) = 0.0
              adist(iw,imod,iact,ishp) = 0.0
           end do
        end do
     end do
  end do
  do iact = 1, nact2
     do iitem = 1, nitem
        do imod = 1, nmod
           do iw = 1, nw
              bdols(iw,imod,iitem,iact) = 0.
              bdist(iw,imod,iitem,iact) = 0.
              bmix(imod,iact) = 0.0
           end do
        end do
     end do
  end do
  do iact = 1, nact2
     do iitem = 1, nitem
        do imod = 1, nmod
           do iw = 1, nw
              cdist(iw,imod,iitem,iact) = 0.
           end do
        end do
     end do
  end do
  do iact = 1, nact2
     do iitem = 1, nitem
        do imod = 1, nmod
           do iw=1,nw
              cdols(iw,imod,iitem,iact) = 0.
           end do
        end do
     end do
  end do
  do iitem = 1, nitem
     do imod = 1, nmod
        ddols(imod,iitem) = 0.
     end do
  end do
  do iact = 1, nact2
     do icon = 1, ncon
```

```
         do imod = 1, nmod
            do iw = 1, nw
               fdols(iw,imod,icon,iact) = 0.
               fdist(iw,imod,icon,iact) = 0.
            end do
         end do
      end do
   end do
   do icsi = 1, ncsi
      do icon = 1, ncon
         do imod = 1, nmod
            gdols(imod,icon,icsi) = 0.
         end do
      end do
   end do
   do iact = 1, nact2
      do icon = 1, ncon
         do imod = 1, nmod
            do iw = 1, nw
               gdist(iw,imod,icon,iact) = 0.
            end do
         end do
      end do
   end do
   do icon = 1, ncon
      do imod = 1, nmod
         hdols(imod,icon) = 0.
      end do
   end do
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw
            result(iw,imod,iact) = 0.
         end do
      end do
   end do
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw
            resulta(iw,imod,iact) = 0.
         end do
      end do
   end do
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw
            resultb(iw,imod,iact) = 0.
         end do
      end do
   end do
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw
            resultf(iw,imod,iact) = 0.
         end do
      end do
   end do
   do iact = 1, nact2
      do imod = 1, nmod
         do iw = 1, nw
            work(iw,imod,iact) = 0.
            resultj(iw,imod,iact) = 0.
         end do
      end do
   end do
   do imod = 1, nmod
      jdols(imod) = 0.
   end do
   print*, 'Matrices initialized '

   open(25,file='nonmods_mp00by_new.dat',recl=1200) !  Non-MODS offices mail proc IOCS data
1  format(a1167,f15.5,i2,i2,i3,i5)

   nt = 0
   er = 0
   tot_dol = 0.0
   tot_dol2 = 0.0
   ct_nowgt = 0

   do while (ier.eq.0)
```

```
      read(25,31,iostat=ier,end=100) rec,dlrs,pool,ldc,iw,actv

      cnt = cnt + 1
      iw = 1

      modgrp = searchi(poolcode,nmod,pool)

      read(f9250,'(f10.0)') rf9250
      read(f166,'(i2)') if166
      read(f167,'(i2)') if167

   Calculation of overhead factors to convert tally dollar weights to cost pool dollars
      gfy = 4833549./4402680.
      ovhfact = 2553568./2328659.
      ovh6522 = 4402680./4340556.
      wgt = rf9250/100000.

      if (actv.ne.6521) then
         wgt6521 = wgt6521+wgt
         wgtall = wgtall+wgt
      else
         wgtall = wgtall + wgt
      end if

      dlrs = wgt*gfy*ovhfact*ovh6522

      if ((modgrp.lt.begmod).or.(modgrp.gt.nmod)) then !  Exclude break tallies
         goto 99
      end if

      tot_dol = tot_dol + dlrs

   Break out Std A ECR Saturation and High Density into separate activity codes
      if ((actv.eq.1311).or.(actv.eq.2311).or.(actv.eq.3311).or.(actv.eq.4311)) then !  Std A  WSH/WSS
         if (f9619.eq.'1') then !  WSS
            if (actv.eq.1311) actv = 1313
            if (actv.eq.2311) actv = 2313
            if (actv.eq.3311) actv = 3313
            if (actv.eq.4311) actv = 4313
         end if
      end if

      if ((actv.eq.1331).or.(actv.eq.2331).or.(actv.eq.3331).or.(actv.eq.4331)) then !  Std A NP WSH/WSS
         if (f9619.eq.'1') then !  WSS
            if (actv.eq.1331) actv = 1333
            if (actv.eq.2331) actv = 2333
            if (actv.eq.3331) actv = 3333
            if (actv.eq.4331) actv = 4333
         end if
      end if

   Any "auto" ECR flats or parcels are assumed to be basic ECR
      if (actv.eq.2312) actv = 2310
      if (actv.eq.3312) actv = 3310
      if (actv.eq.4312) actv = 4310
      if (actv.eq.2332) actv = 2330
      if (actv.eq.3332) actv = 3330
      if (actv.eq.4332) actv = 4330

      deals with new international mixed codes

      ishp = shapeind(actv,f9635,f9805) !  Subroutine assigns shape

   Assign handling category
      if (((actv.ge.1000).and.(actv.le.4950)).or.((actv.ge.5300).and.(actv.le.5480))) then
         hand = 1             ! direct (non special services)
      else if ((actv.ge.10).and.(actv.lt.1000)) then
         if (((f9805.ge.'1000').and.(f9805.le.'4950')).or.
     &       ((f9805(1:2).ge.'53').and.(f9805(1:2).le.'54')))then
            hand = 1          ! direct (non special services)
         else if ((f9635.ge.'A').and.(f9635.le.'K')) then
            hand = 1          ! direct (special services)
         else if ((f9214.ge.'A').and.(f9214.le.'P')) then
            hand = 2          ! mixed item
         else if ((f9219.ge.'A').and.(f9219.le.'J')) then
            hand = 3          ! mixed container
         else
            hand = 4          ! not handling mail
         end if
```

```
      else if ((f9214.ge.'A').and.(f9214.le.'P')) then
         hand = 2             ! mixed item
      else if ((f9219.ge.'A').and.(f9219.le.'J')) then
         hand = 3             ! mixed container
      else
         hand = 4             ! not handling mail
      end if

      iitem = searchc(codes,nitem,f9214) !  Assign item type
      icon = searchc(codes,ncon,f9219) !  Assign container type
      iact = searchi(acodes,nact2,actv) !  Activity codes

C     Assign weight increment
      if (hand.eq.1) then
         if (actv.ge.1000) then
            iw = weight(f165,if166,if167,ct_nowgt,nw) !  Subroutine assigns weight increment
         else
            iw = nw           !  Special service activities assumed to have no record weight
         end if
      else
         iw = nw
      end if

      if ((hand.eq.1).and.(iact.eq.0)) then
         print *,'missing direct activity code = ',actv,' modgrp = ',modgrp
      end if

C     Single piece being handled, Assign to A matrix
      if ((hand.eq.1).and.(((iitem.eq.0).and.(icon.eq.0).and.(f9213.eq.'A'))
     &     .or.(f129.eq.'B'))) then
         if (iact.gt.0) then
            if ((modgrp.gt.0).and.(modgrp.le.nmod)) then
               adols(iw,modgrp,iact,ishp)=adols(iw,modgrp,iact,ishp) + dlrs
               atot = atot + dlrs
               acnt = acnt + 1
               tot_dol2 = tot_dol2 + dlrs
            else
               print *,' bad MODS in matrix A ',f114, modgrp, dlrs
            end if
         else                 !  Not handling mail
            print *,'Not-handling tally with direct code = ',actv,' cost pool = ',modgrp
            if (modgrp.gt.0) then
               jdols(modgrp) = jdols(modgrp) + dlrs
               jtot = jtot + dlrs
               jcnt = jcnt + 1
               tot_dol2 = tot_dol2 + dlrs
            end if
         end if

C*****************************************************************************
C     Not-handling mail tallies -- assign to J matrix
      else if (hand.eq.4) then

         jdols(modgrp) = jdols(modgrp) + dlrs
         jtot = jtot + dlrs
         jcnt = jcnt + 1
         tot_dol2 = tot_dol2 + dlrs

C*****************************************************************************
C   Item being handled: separate items with direct activity codes from others

      else if ((f9214.ge.'A').and.(f9214.le.'P')) then
         if (hand.eq.1) then
            imat = 1          ! "B" matrix - identical, top piece, or counted item
         else if (hand.eq.2) then
            imat = 3          ! "D" matrix - mixed, empty item
         else
            print *,'problem item in modgrp = ',modgrp
            imat = 0
         end if

C     "D" matrix: mixed or empty item
         if (imat.eq.3) then
            ddols(modgrp,iitem) = ddols(modgrp,iitem) + dlrs
            dtot = dtot + dlrs
            dcnt = dcnt + 1
            tot_dol2 = tot_dol2 + dlrs

C     "B" matrix: identical or top piece rule (direct item)
         else if (imat.eq.1) then
```

```
            bdols(iw,modgrp,iitem,iact) =
     &         bdols(iw,modgrp,iitem,iact) + dlrs
            btot = btot + dlrs
            bcnt = bcnt + 1
            tot_dol2 = tot_dol2 + dlrs
         else                  !  Not handling mail
            print *,' imat 0 in modgrp = ',actv
            jdols(modgrp) = jdols(modgrp) + dlrs
            jtot = jtot + dlrs
            jcnt = jcnt + 1
            tot_dol2 = tot_dol2 + dlrs
         end if

C*****************************************End Item*****************************
C     Container being handled: separate containers with direct activity codes from others
      else if (icon.gt.0) then

         if (modgrp.gt.0) then

            flag2=.false.

            if (f9901(1:1).eq.'%') then
               read(rec(340:406),451,iostat=ier) counts
            else
               read(rec(339:406),450,iostat=ier) counts
            end if
450         format(5(1x,f3.0),16f3.0)
451         format(f3.0,4(1x,f3.0),16f3.0)


            if (ier.ne.0) then
               flag2 = .true.
               j = 340
               do i = 1, ncsi
                  counts(i) = 0.
               end do
               ier = 0
            end if

            sum = 0.
            do i = 1, ncsi
               sum = sum + counts(i)
            end do

c     "F" matrix: identical mail in container (direct container)
            if (hand.eq.1) then
               fdols(iw,modgrp,icon,iact) =
     &            fdols(iw,modgrp,icon,iact) + dlrs
               ftot = ftot + dlrs
               fcnt = fcnt + 1
               tot_dol2 = tot_dol2 + dlrs

c     "H" matrix:  Uncounted, empty, or contents read error
            else if ((sum.eq.0.).or.flag2) then
               hdols(modgrp,icon) = hdols(modgrp,icon) + dlrs
               htot = htot + dlrs
               hcnt = hcnt + 1
               tot_dol2 = tot_dol2 + dlrs

c     "G" matrix:  container contents are "identified"
            else if (sum.gt.0.) then
               do icsi = 1, ncsi
                  gdols(modgrp,icon,icsi) = gdols(modgrp,icon,icsi) +
     &               (counts(icsi)/sum) * dlrs
               end do
               gtot = gtot + dlrs
               gcnt = gcnt + 1
               tot_dol2 = tot_dol2 + dlrs
            end if
         else                  !  Not handling
            print *,' bad container or mods code ',f9219,',',modgrp
            jdols(modgrp) = jdols(modgrp) + dlrs
            jtot = jtot + dlrs
            jcnt = jcnt + 1
            tot_dol2 = tot_dol2 + dlrs
         end if

C************************************End Container*****************************
c     Any remaining tallies considered not handling mail
      else
```

```
            jdols(modgrp) = jdols(modgrp) + dlrs
            jtot = jtot + dlrs
            jcnt = jcnt + 1
            tot_dol2 = tot_dol2 + dlrs
         end if

99       id do
100   print *,' read exit = ',ier,' with ',cnt,' records ', ' dlrs = ', tot_dol
      print*, 'Total assigned dlrs = ', tot_dol2

C ***********************************End Read Loop*********************************
c     Redistribute no weight direct single piece costs

      do iact = begmail, nact2
         do imod = 1, nmod
            do ishp = 1, nshp
               if (adols(nw,imod,iact,ishp).gt.0.0) then
                  sum = 0.0
                  do iw = 1, nw2 !  Distribute over all weight increments
                     sum = sum + adols(iw,imod,iact,ishp)
                  end do
                  if (sum.gt.0) then
                     do iw = 1, nw2
                        adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                     adols(nw,imod,iact,ishp)*adols(iw,imod,iact,ishp)/sum
                     end do
                     adols(nw,imod,iact,ishp) = 0.0
                  end if
               end if
            end do
         end do
      end do

c     Residual distribution of direct single piece no weight costs

      do iact = begmail, nact2
         do imod = 1, nmod
            do ishp = 1, nshp
               if (adols(nw,imod,iact,ishp).gt.0.0) then
                  sum = 0.0
                  do iw = 1, nw2
                     actwgt(iw) = 0.0
                  end do
                  do j = 1, nmod !  Distbribute over all cost pools
                     do iw = 1, nw2 !  Distribute over all weight increments
                        actwgt(iw) = actwgt(iw) + adols(iw,j,iact,ishp)
                        sum = sum + adols(iw,j,iact,ishp)
                     end do
                  end do
                  if (sum.gt.0) then
                     do iw = 1, nw2
                        adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                     adols(nw,imod,iact,ishp)*actwgt(iw)/sum
                     end do
                     adols(nw,imod,iact,ishp) = 0.0
                  else
                     if (adols(nw,imod,iact,ishp).gt.0.) then
                        print*,'Level 3 distribution of act = ',acodes(iact)
                        do k = begmail, nact2
                           do iw = 1, nw2
                              actshr2(iw,k) = 0.
                           end do
                        end do
                        do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                           if (class(k).eq.class(iact)) then !  Same subclass
                              do iw = 1, nw2 !  Distribute over all weight increments
                                 actshr2(iw,k) = actshr2(iw,k) + adols(iw,imod,k,ishp)
                                 sum = sum + adols(iw,imod,k,ishp)
                              end do
                           end if
                        end do
                        if (sum.gt.0.) then
                           do k = begmail, nact2
                              do iw = 1, nw2
                                 adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                              adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                              end do
                           end do
                           adols(nw,imod,iact,ishp) = 0.0
                        else
```

```
                        print*,'Level 4 distribution of act = ',acodes(iact)
                        do k = begmail, nact2
                           do iw = 1, nw2
                              actshr2(iw,k) = 0.
                           end do
                        end do
                        do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                           if (class(k).eq.class(iact)) then !  Same subclass
                              do j = 1,nmod !  Distribute over all cost pools
                                 do iw = 1, nw2 !  Disbribute over all weight increments
                                    actshr2(iw,k) = actshr2(iw,k) + adols(iw,j,k,ishp)
                                    sum = sum + adols(iw,j,k,ishp)
                                 end do
                              end do
                           end if
                        end do
                        if (sum.gt.0.) then
                           do k = begmail, nact2
                              do iw = 1, nw2
                                 adist(iw,imod,iact,ishp) = adist(iw,imod,iact,ishp) +
     &                              adols(nw,imod,iact,ishp) * actshr2(iw,k) / sum
                              end do
                           end do
                           adols(nw,imod,iact,ishp) = 0.0
                        else
                           print*,'unable to distribute no weight for ',
     &                        imod,' act = ',acodes(iact), ' cost = ', adols(nw,imod,iact,ishp)
                        end if
                     end if
                  end if
               end if
            end do
         end do
      end do

c     Add in redistributed no weight direct single piece costs
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               do ishp = 1, nshp
                  adols(iw,imod,iact,ishp) = adols(iw,imod,iact,ishp) + adist(iw,imod,iact,ishp)
               end do
            end do
         end do
      end do

c     Redistribute no weight identical/top piece item costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do iitem = 1, nitem
               if (bdols(nw,imod,iitem,iact).gt.0) then
                  sum = 0.0
                  do iw = 1, nw2 !  Distribute over all weight increments
                     sum = sum + bdols(iw,imod,iitem,iact)
                  end do
                  if (sum.gt.0.0) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
     &                     bdols(nw,imod,iitem,iact)*bdols(iw,imod,iitem,iact)/sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  end if
               end if
            end do
         end do
      end do

c     Residual distribution of identical/top piece items no weight costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do iitem = 1, nitem
               if (bdols(nw,imod,iitem,iact).gt.0.0) then
                  sum = 0.0
                  do iw = 1, nw2
                     actwgt(iw) = 0.0
                  end do
                  do j = 1, nitem !  Distribute over all item types
                     do iw = 1, nw2 !  Distribute over all weight increments
                        actwgt(iw) = actwgt(iw) + bdols(iw,imod,j,iact)
```

```
            sum = sum + bdols(iw,imod,j,iact)
         end do
      end do
      if (sum.gt.0) then
         do iw = 1, nw2
            bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
               bdols(nw,imod,iitem,iact)*actwgt(iw)/sum
         end do
         bdols(nw,imod,iitem,iact) = 0.0
      else
         if (bdols(nw,imod,iitem,iact).gt.0.0) then
            print*, 'Level 3 b distribution of act = ', acodes(iact)
            do iw = 1, nw2
               actwgt(iw) = 0.
            end do
            do k = begmail, nmod !  Distribute over all cost pools
               do j = 1,nitem !  Distribute over all item types
                  do iw = 1, nw2 !  Distribute over all weight increments
                     actwgt(iw) = actwgt(iw) + bdols(iw,k,j,iact)
                     sum = sum + bdols(iw,k,j,iact)
                  end do
               end do
            end do
            if (sum.gt.0.) then
               do iw = 1, nw2
                  bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                    bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
               end do
               bdols(nw,imod,iitem,iact) = 0.0
            else
               print*, 'Level 4 b distribution of act = ', acodes(iact)
               do iw = 1, nw2
                  actwgt(iw) = 0.
               end do
               do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                  if (class(k).eq.class(iact)) then !  Same subclass
                     do j = 1, nmod !  Distribute over all cost pools
                        do l = 1,nitem !  Distribute over all item types
                           do iw = 1, nw2 !  Disbribute over all weight increments
                              actwgt(iw) = actwgt(iw) + bdols(iw,j,l,k)
                              sum = sum + bdols(iw,j,l,k)
                           end do
                        end do
                     end do
                  end if
               end do
               if (sum.gt.0.) then
                  do iw = 1, nw2
                     bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                       bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                  end do
                  bdols(nw,imod,iitem,iact) = 0.0
               else
                  print*, 'Level 5 b distribution of act = ', acodes(iact)
                  do iw = 1, nw2
                     actwgt(iw) = 0.
                  end do
                  do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                     if (class_code(k).eq.class_code(iact)) then !  Same subclass
                        do j = 1, nmod !  Distribute over all cost pools
                           do l = 1,nitem !  Distribute over all item types
                              do iw = 1, nw2 !  Disbribute over all weight increments
                                 actwgt(iw) = actwgt(iw) + bdols(iw,j,l,k)
                                 sum = sum + bdols(iw,j,l,k)
                              end do
                           end do
                        end do
                     end if
                  end do
                  if (sum.gt.0.) then
                     do iw = 1, nw2
                        bdist(iw,imod,iitem,iact) = bdist(iw,imod,iitem,iact) +
&                          bdols(nw,imod,iitem,iact) * actwgt(iw) / sum
                     end do
                     bdols(nw,imod,iitem,iact) = 0.0
                  else
                     print*,'unable to distribute no weight for b, ',
&                       imod,' act = ',acodes(iact), ' cost = ', bdols(nw,imod,iitem,iact)
                  end if
               end if
```

```
                  end if
               end if
            end if
         end if
      end do
   end do
id do

c     Add in redistributed no weight identical/top piece item costs
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               do iitem = 1, nitem
                  bdols(iw,imod,iitem,iact) = bdols(iw,imod,iitem,iact) + bdist(iw,imod,iitem,iact)
                  bdist(iw,imod,iitem,iact) = 0.0
               end do
            end do
         end do
      end do

c     Residual distribution of identical/top piece container no weight costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do icon = 1, ncon
               if (fdols(nw,imod,icon,iact).gt.0) then
                  sum = 0.0
                  do iw = 1, nw2 !  Distribute over all weight increments
                     sum = sum + fdols(iw,imod,icon,iact)
                  end do
                  if (sum.gt.0.0) then
                     do iw = 1, nw2
                        fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &                     fdols(nw,imod,icon,iact)*fdols(iw,imod,icon,iact)/sum
                     end do
                     fdols(nw,imod,icon,iact) = 0.0
                  end if
               end if
            end do
         end do
nd do

c     Residual distribution of identical/top piece container no weight costs
      do iact = begmail, nact2
         do imod = 1, nmod
            do icon = 1, ncon
               if (fdols(nw,imod,icon,iact).gt.0.0) then
                  sum = 0.0
                  check = 0.0
                  do iw = 1, nw2
                     actwgt(iw) = 0.0
                  end do
                  do j = 1, nmod !  Distbribute over all cost pools
                     do iw = 1, nw2 !  Distribute over all weight increments
                        actwgt(iw) = actwgt(iw) + fdols(iw,j,icon,iact)
                        sum = sum + fdols(iw,j,icon,iact)
                     end do
                  end do
                  if (sum.gt.0) then
                     do iw = 1, nw2
                        fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &                     fdols(nw,imod,icon,iact)*actwgt(iw)/sum
                     end do
                     fdols(nw,imod,icon,iact) = 0.0
                  else
                     if (fdols(nw,imod,icon,iact).gt.0.) then
                        print*,'Level 3 distribution of f act = ',acodes(iact)
                        do k = begmail, nact2
                           do iw = 1, nw2
                              actshr2(iw,k) = 0.
                           end do
                        end do
                        do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                           if (class(k).eq.class(iact)) then !  Same subclass
                              do iw = 1, nw2 !  Distribute over all weight increments
                                 actshr2(iw,k) = actshr2(iw,k) + fdols(iw,imod,icon,k)
                                 sum = sum + fdols(iw,imod,icon,k)
                              end do
                           end if
                        end do
                        if (sum.gt.0.) then
```

```
                        do k = begmail, nact2
                           do iw = 1, nw2
                              fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &                           fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                           end do
                        end do
                        fdols(nw,imod,icon,iact) = 0.0
                     else
                        print*,'Level 4 distribution f of act = ',acodes(iact)
                        do k = begmail, nact2
                           do iw = 1, nw2
                              actshr2(iw,k) = 0.
                           end do
                        end do
                        do k = begmail, nact2 !  Distribute over all activity codes within same subclass
                           if (class(k).eq.class(iact)) then !  Same subclass
                              do j = 1,nmod !  Distribute over all cost pools
                                 do iw = 1, nw2 !  Distribute over all weight increments
                                    actshr2(iw,k) = actshr2(iw,k) + fdols(iw,j,icon,iact)
                                    sum = sum + fdols(iw,j,icon,iact)
                                 end do
                              end do
                           end if
                        end do
                        if (sum.gt.0.) then
                           do k = begmail, nact2
                              do iw = 1, nw2
                                 fdist(iw,imod,icon,iact) = fdist(iw,imod,icon,iact) +
     &                              fdols(nw,imod,icon,iact) * actshr2(iw,k) / sum
                              end do
                           end do
                           fdols(nw,imod,icon,iact) = 0.0
                        else
                           print*,'unable to distribute no weight f for ',
     &                        imod,' act = ',acodes(iact), ' cost = ', fdols(nw,imod,icon,iact)
                        end if
                     end if
                  end if
               end if
            end if
         end do
      end do
   end do

c     Add in redistributed no weight identical/top piece container costs
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               do icon = 1, ncon
                  fdols(iw,imod,icon,iact) = fdols(iw,imod,icon,iact) + fdist(iw,imod,icon,iact)
                  fdist(iw,imod,icon,iact) = 0.0
               end do
            end do
         end do
      end do

c     Distribute mixed/empty item costs ("D" matrix) using direct item costs ("B" matrix) as a distribution key
c     over all activity codes and weight increments within cost pool and item type

      do imod = begmod, nmod
         do iitem = 1, nitem
            if (ddols(imod,iitem).gt.0.) then
               sum = 0.
               do iact = 1, nact2 !  Distribute over all activity code
                  do iw = 1, nw !  Distribute over all weight increments
                     sum = sum + bdols(iw,imod,iitem,iact)
                  end do
               end do
               if (sum.gt.0) then
                  do iact = 1, nact2
                     do iw = 1, nw
                        cdist(iw,imod,iitem,iact) = cdist(iw,imod,iitem,iact) +
     &                     ddols(imod,iitem) * bdols(iw,imod,iitem,iact) / sum
                     end do
                  end do
                  ddols(imod,iitem) = 0.
               end if
            end if
         end do
      end do
```

```
c     Distribute remaining mixed/empty item costs ("D" matrix) over all activity codes, weight increments,
C     and cost pools within item type using direct item costs ("B" matrix)
      do iitem = 1, nitem
         do imod = begmod, nmod
            if (ddols(imod,iitem).gt.0.) then
               print *,'residual distribution of item = ',iitem,' pool = ',imod
               sum = 0
               do iact = 1, nact2
                  do iw = 1, nw
                     actshr(iw,iact) = 0.
                  end do
               end do
               do iact = 1, nact2 !  Distribute over all activity codes
                  do j = begmod, nmod !  Distribute over all cost pools
                     do iw = 1, nw !  Distribute over all weight increments
                        actshr(iw,iact) = actshr(iw,iact) + bdols(iw,j,iitem,iact)
                        sum = sum + bdols(iw,j,iitem,iact)
                     end do
                  end do
               end do
               if (sum.gt.0.) then
                  do iact = 1, nact2
                     do iw = 1, nw
                        cdist(iw,imod,iitem,iact) = cdist(iw,imod,iitem,iact) +
     &                     ddols(imod,iitem) * actshr(iw,iact) / sum
                     end do
                  end do
               else
                  print *,' unable to dist D dols for iitem = ',iitem,', ',ddols(imod,iitem)
               end if
            end if
         end do
      end do

C     Distribute "identified" container costs ("G" matrix)

      do imod = begmod, nmod    !  Initial distribution within cost pools

         if (imod.ne.1) then    !  Exclude Allied pool

            do icsi = 1, ncsi
               if (icsi.le.5) then ! Loose shapes distributed based upon direct piece costs ("A" matrix)
                  sum = 0.
                  distsum = 0.
                  do iact = 1, nact2 !  Distribute over all activity codes
                     do iw = 1, nw !  Distribute over all weight increments
                        sum = sum + adols(iw,imod,iact,icsi)
                     end do
                  end do
                  if (sum.gt.0.) then
                     do icon = 1, ncon
                        if (gdols(imod,icon,icsi).gt.0.) then
                           do iact = 1, nact2
                              do iw = 1, nw
                                 gdist(iw,imod,icon,iact) =
     &                              gdist(iw,imod,icon,iact) +
     &                              gdols(imod,icon,icsi) *
     &                              adols(iw,imod,iact,icsi) / sum
                              end do
                           end do
                        end if
                     end do
                  else          ! distribute over all cost pools, activity codes, and weight increments
                     do iact = 1, nact2 !  Distribute over all activity codes
                        do i = begmod, nmod !  Distribute over all cost pools
                           do iw = 1 , nw !  Distribute over all weight increments
                              distsum = distsum + adols(iw,i,iact,icsi)
                           end do
                        end do
                     end do
                     if (distsum.gt.0) then
                        do iact = 1, nact2
                           do icon = 1, ncon
                              do i = begmod, nmod
                                 do iw = 1, nw
                                    gdist(iw,imod,icon,iact) =
     &                                 gdist(iw,imod,icon,iact) +
     &                                 gdols(imod,icon,icsi) *
     &                                 adols(iw,i,iact,icsi)/distsum
```

```
                        end do
                     end do
                  end do
               end do
            else
               print *,'shape distribution empty: mod = ',imod,
                  ', shape = ',icsi
            end if
         end if
      else              ! Items distributed upon direct item costs ("B" matrix)
         iitem = icsi - nshp2
         sum = 0.
         distsum = 0.
         do iact = 1, nact2 !  Distribute over all activity codes
            do iw = 1, nw ! Distribute over all weight increments
               sum = sum + bdols(iw,imod,iitem,iact)
            end do
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do iact = 1, nact2
                     do iw = 1, nw
                        gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
&                          gdols(imod,icon,icsi) * bdols(iw,imod,iitem,iact) / sum
                     end do
                  end do
               end if
            end do
         else          ! distribute over all cost pools, activity codes, and weight increments
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = begmod, nmod !  Distribute over all cost pools
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + bdols(iw,i,iitem,iact)
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
                     do i = begmod, nmod
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) =
&                             gdist(iw,imod,icon,iact) +
&                             gdols(imod,icon,icsi) *
&                             bdols(iw,i,iitem,iact)/distsum
                        end do
                     end do
                  end do
               end do
            else
               check = 0.
               do icon = 1,ncon
                  check = check + gdols(imod,icon,icsi)
               end do
               if (check.gt.0.) then
                  print *,'shape distribution empty: mod = ',imod,
&                    ', shape = ',icsi
               end if
            end if
         end if
      end if
   end do

else if (imod.eq.1) then ! Distribute allied over all cost pools, activity codes, and weight increments

   do icsi = 1, ncsi
      if (icsi.le.5) then ! Loose shapes distributed based upon direct piece costs ("A" matrix)
         sum = 0.
         distsum = 0.
         do i = begmod, nmod !  Distribute over all cost pools
            if ((i.ne.7).and.(i.ne.8)) then !  Exclude Registry and Misc
               do iact = 1, nact2 !  Distribute over all activity codes
                  do iw = 1, nw !  Distribute over all weight increments
                     sum = sum + adols(iw,i,iact,icsi)
                  end do
               end do
            end if
         end do
         if (sum.gt.0.) then
```

```
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do i = begmod, nmod
                     if ((i.ne.7).and.(i.ne.8)) then !  Exclude Registry and Misc
                        do iact = 1, nact2
                           do iw = 1, nw
                              gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
&                                gdols(imod,icon,icsi) * adols(iw,i,iact,icsi) / sum
                           end do
                        end do
                     end if
                  end do
               end if
            end do
         else          ! distribute over all cost pools, activity codes, and weight increments
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = begmod, nmod !  Distribute over all cost pools
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + adols(iw,i,iact,icsi)
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
                     do i = begmod, nmod
                        do iw = 1, nw
                           gdist(iw,imod,icon,iact) =
&                             gdist(iw,imod,icon,iact) +
&                             gdols(imod,icon,icsi) *
&                             adols(iw,i,iact,icsi)/distsum
                        end do
                     end do
                  end do
               end do
            else
               print *,'shape distribution empty: mod = ',imod,
&                 ', shape = ',icsi
            end if
         end if
      else             ! Allied items distributed upon identical/top piece item costs ("B" matrix)
         iitem = icsi - nshp2
         sum = 0.
         distsum = 0.
         do i = begmod, nmod !  Distribute over all cost pools
            if ((i.ne.7).and.(i.ne.8)) then !  Exclude Registry and Misc
               do iact = 1, nact2 !  Distribute over all activity codes
                  do iw = 1, nw !  Distribute over all weight increments
                     sum = sum + bdols(iw,i,iitem,iact)
                  end do
               end do
            end if
         end do
         if (sum.gt.0.) then
            do icon = 1, ncon
               if (gdols(imod,icon,icsi).gt.0.) then
                  do i = begmod, nmod
                     if ((i.ne.7).and.(i.ne.8)) then !  Exclude Registry and Misc
                        do iact = 1, nact2
                           do iw = 1, nw
                              gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
&                                gdols(imod,icon,icsi) * bdols(iw,i,iitem,iact) / sum
                           end do
                        end do
                     end if
                  end do
               end if
            end do
         else          ! distribute over all cost pools, activity codes, and weight increments
            print *,'Allied level 2 item = ',iitem
            do iact = 1, nact2 !  Distribute over all activity codes
               do i = begmod, nmod !  Distribute over all cost pools
                  do iw = 1 , nw !  Distribute over all weight increments
                     distsum = distsum + bdols(iw,i,iitem,iact)
                  end do
               end do
            end do
            if (distsum.gt.0) then
               do iact = 1, nact2
                  do icon = 1, ncon
```

```
                        do i = begmod, nmod
                           do iw = 1, nw
                              gdist(iw,imod,icon,iact) =
     &                           gdist(iw,imod,icon,iact) +
                                 gdols(imod,icon,icsi) *
                                 bdols(iw,i,iitem,iact)/distsum
                           end do
                        end do
                     end do
                  end do
               else
                  print *,'item distribution empty: mod = ',imod,
     &                 ', item = ',icsi
               end if
            end if
         end if

      end do

   end if
end do                    ! End of "identified" container ("G" matrix) distribution

c  Additional item and container distributions
   do iact = 1,nact2
      do iw = 1,nw
         do iitem = 1,nitem
            cdist(iw,4,iitem,iact) = cdist(iw,4,iitem,iact)*23135.8/(23135.8-218.8) !  Manual Flats
            cdist(iw,6,iitem,iact) = cdist(iw,6,iitem,iact)*9539.6/(9539.6-942.6) !  Manual Parcels
            cdist(iw,8,iitem,iact) = cdist(iw,8,iitem,iact)*5823.7/(5823.7-199.7) !  Misc
         end do
         do icon = 1,ncon
            gdist(iw,1,icon,iact) = gdist(iw,1,icon,iact)*52293.4/(52293.4-686.7) !  Allied
            gdist(iw,4,icon,iact) = gdist(iw,4,icon,iact)*23135.8/(23135.8-218.8) !  Manual Flats
            gdist(iw,6,icon,iact) = gdist(iw,6,icon,iact)*9539.6/(9539.6-942.6) !  Manual Parcels
            gdist(iw,8,icon,iact) = gdist(iw,8,icon,iact)*5823.7/(5823.7-199.7) !  Misc
         end do
      end do
   end do

c   um distributed "identified" container costs into direct container costs ("F" matrix)
   do iact = 1, nact2
      do icon = 1, ncon
         do imod = begmod, nmod
            do iw = 1, nw
               fdols(iw,imod,icon,iact) = fdols(iw,imod,icon,iact) + gdist(iw,imod,icon,iact)
               gdist(iw,imod,icon,iact) = 0.
            end do
         end do
      end do
   end do

c  Distribute uncounted/empty containers ("H" matrix) using direct and distributed "identified"
c  container costs ("F" matrix) over all activity codes and weight increment categories within cost pool and
c  container type

   do imod = begmod, nmod
      do icon = 1, ncon
         sum = 0.
         do iact = 1, nact2  !  Distribute over all activity codes
            do iw = 1, nw    !  Distribute over all weight increments
               sum = sum + fdols(iw,imod,icon,iact)
            end do
         end do
         if (sum.gt.0) then
            do iact = 1, nact2
               do iw = 1, nw
                  gdist(iw,imod,icon,iact) =
     &                 hdols(imod,icon) * fdols(iw,imod,icon,iact) / sum
               end do
            end do
            hdols(imod,icon) = 0.
         end if
      end do
   nd do

   Distribute remaining uncounted/empty container costs ("H" matrix) over all activity codes, weight
   increments, and cost pools within container type using direct/distributed "identified" container
   costs ("F" matrix)

   do icon = 1, ncon
```

```
      do imod = begmod, nmod
         if (hdols(imod,icon).gt.0.) then
            sum = 0.
            do iact = 1, nact2
               do iw = 1, nw
                  actshr(iw,iact) = 0.
               end do
            end do
            do iact = 1, nact2 !  Distribute over all activity codes
               do j = begmod, nmod !  Distribute over all cost pools
                  do iw = 1, nw !  Distribute over all weight increments
                     actshr(iw,iact) = actshr(iw,iact) + fdols(iw,j,icon,iact)
                     sum = sum + fdols(iw,j,icon,iact)
                  end do
               end do
            end do
            if (sum.gt.0.) then
               do iact = 1, nact2
                  do iw = 1, nw
                     gdist(iw,imod,icon,iact) = gdist(iw,imod,icon,iact) +
     &                  actshr(iw,iact)/sum * hdols(imod,icon)
                  end do
               end do
            else
               print *,' unable to dist h dols for imod = ',imod,
     &              ' icon = ',icon
            end if
         end if
      end do
   end do

c     Sum up all costs (direct and redistributed) except not handling costs ("J" matrix)
c     Pieces
      do ishp = 1, nshp
         do iact = 1, nact2
            do imod = begmod, nmod
               do iw = 1, nw
                  result(iw,imod,iact) = result(iw,imod,iact) + adols(iw,imod,iact,ishp)
                  resulta(iw,imod,iact) = resulta(iw,imod,iact) + adols(iw,imod,iact,ishp)
               end do
            end do
         end do
      end do
c     Items
      do iact = 1, nact2
         do iitem = 1, nitem
            do imod = begmod, nmod
               do iw = 1, nw
                  result(iw,imod,iact) = result(iw,imod,iact) + bdols(iw,imod,iitem,iact)
     &                 + cdols(iw,imod,iitem,iact) + cdist(iw,imod,iitem,iact)
                  resultb(iw,imod,iact) = resultb(iw,imod,iact) + bdols(iw,imod,iitem,iact)
     &                 + cdols(iw,imod,iitem,iact) + cdist(iw,imod,iitem,iact)
               end do
            end do
         end do
      end do
c     Containers
      do iact = 1, nact2
         do icon = 1, ncon
            do imod = begmod, nmod
               do iw = 1, nw
                  result(iw,imod,iact) = result(iw,imod,iact) + fdols(iw,imod,icon,iact) +
     &                 gdist(iw,imod,icon,iact)
                  resultf(iw,imod,iact) = resultf(iw,imod,iact) + fdols(iw,imod,icon,iact) +
     &                 gdist(iw,imod,icon,iact)
               end do
            end do
         end do
      end do

C     Distribute not handling costs ("J" matrix) using all other costs ("results" matrix)
      do imod = begmod, nmod
         sum = 0.
         distsum = 0.
         if ((imod.ne.1).and.(imod.ne.8)) then !  Exclude allied and miscellaneous cost pools
            do iact = 1, nact2  !  Distribute over all activity codes
               do iw = 1, nw    !  Distribute over all weight increments
                  sum = sum + result(iw,imod,iact)
               end do
            end do
```

```
         if (sum.gt.0) then
            do iact = 1, nact2
               do iw = 1, nw
                  work(iw,imod,iact) = work(iw,imod,iact) +
                     jdols(imod) * result(iw,imod,iact) / sum

               end do
            end do
         else
            print *,' unable to distribute J dollars for ',imod
         end if

      else if (imod.eq.1) then !  Allied cost pool
         do iact = 1, nact2  !  Distribute over all activity codes
            do i = begmod, nmod !  Distribute over all cost pools
               if ((i.ne.7).and.(i.ne.8)) then !  Exclude Registry and Misc cost pools
                  do iw = 1, nw !  Distribute over all weight increments
                     distsum = distsum + result(iw,i,iact)
                  end do
               end if
            end do
         end do
         if (distsum.gt.0) then
            do iact = 1, nact2
               do i = begmod, nmod
                  if ((i.ne.7).and.(i.ne.8)) then !  Exclude Registry and Misc
                     do iw = 1, nw
                        work(iw,imod,iact) = work(iw,imod,iact) +
     &                     jdols(imod) * result(iw,i,iact) / distsum
                     end do
                  end if
               end do
            end do
         else
            print *,' unable to distribute J dollars for ',imod
         end if

      else if (imod.eq.8) then !  Misc cost pool
         do iact = 1, nact2  !  Distribute over all activity codes
            do i = begmod, nmod !  Distribute over all cost pools
               do iw = 1, nw !  Distribute over all weight increments
                  distsum = distsum + result(iw,i,iact)
               end do
            end do
         end do
         if (distsum.gt.0) then
            do iact = 1, nact2
               do i = begmod, nmod
                  do iw = 1, nw
                     work(iw,imod,iact) = work(iw,imod,iact) +
     &                  jdols(imod) * result(iw,i,iact) / distsum
                  end do
               end do
            end do
         else
            print *,' unable to distribute J dollars for ',imod
         end if

      end if

   end do

   Sum distributed not handling costs ("J" matrix) into handling costs ("results" matrix)
   do iact = 1, nact2
      do imod = begmod, nmod
         do iw = 1, nw
            result(iw,imod,iact) = result(iw,imod,iact) + work(iw,imod,iact)
            resultj(iw,imod,iact) = work(iw,imod,iact)
            work(iw,imod,iact) = 0.
         end do
      end do
   end do

   Redistribute class-specific mixed mail costs over appropriate class-specific direct activity codes,
   weight increments, and within cost pools
   do imod = 1,nmod
      do iact = 1,nmixcl
         do iw = 1, nw
            if (result(iw,imod,nact+iact).gt.0.0) then
               sum = 0.
```

```
            do i = 1,nact
               actshr3(i) = 0.
            end do
            do i = 1,nact !  Distribute over all direct activity codes
               do j = 1,nw !  Distribute over all weight increments
                  if (mixmap(i,iact).gt.0) then
                     sum = sum + result(j,imod,mixmap(i,iact))
                     actshr3(mixmap(i,iact)) = actshr3(mixmap(i,iact))
     &                  + result(j,imod,mixmap(i,iact))
                  end if
               end do
            end do
            if (sum.gt.0.) then
               do i = 1,nact
                  if (mixmap(i,iact).gt.0) then
                     work(iw,imod,mixmap(i,iact)) =
     &                  work(iw,imod,mixmap(i,iact)) +
     &                  (result(iw,imod,nact+iact)*
     &                  actshr3(mixmap(i,iact))/sum)
                  end if
               end do
               result(iw,imod,nact+iact) = 0.
            else
               sum = 0.
               do i = 1,nact
                  actshr3(i) = 0.
               end do
               do i = 1, nact !  Distribute over all direct activity codes
                  do j = 1,nw !  Distribute over all weight increments
                     do k = 1, nmod !  Distribute over all cost pools
                        if (mixmap(i,iact).gt.0) then
                           sum = sum + result(j,k,mixmap(i,iact))
                           actshr3(mixmap(i,iact)) = actshr3(mixmap(i,iact))
     &                        + result(j,k,mixmap(i,iact))
                        end if
                     end do
                  end do
               end do
               if (sum.gt.0.) then
                  do i = 1, nact
                     if (mixmap(i,iact).gt.0) then
                        work(iw,imod,mixmap(i,iact)) =
     &                     work(iw,imod,mixmap(i,iact)) +
     &                     (result(iw,imod,nact+iact)*
     &                     actshr3(mixmap(i,iact))/sum)
                     end if
                  end do
                  result(iw,imod,nact+iact) = 0.
               else
                  print*, 'Mix actv code not distributed ', acodes(nact+iact),
     &                 ' cost = ', result(iw,imod,nact+iact), ' pool ', modcodes(imod)
               end if
            end if
         end if
      end do
   end do
end do

Sum distributed class-specific mixed-mail costs into all other costs
do iact = 1, nact
   do imod = begmod, nmod
      do iw = 1, nw
         result(iw,imod,iact) = result(iw,imod,iact) + work(iw,imod,iact)
         work(iw,imod,iact) = 0.
      end do
   end do
end do

Compute volume-variable costs
distsum = 0.
do imod = begmod, nmod
   sum = 0.
   do iact = 1,nact
      do iw = 1,nw
         sum = sum + result(iw,imod,iact)
      end do
   end do
   if (sum.gt.0.) then
      do iact = 1,nact
         do iw = 1,nw
```

```
                varcost(iw,imod,iact) = varcost(iw,imod,iact) +
     &             result(iw,imod,iact)*variable(imod)
                novarcst(iw,imod,iact) = result(iw,imod,iact)
             end do
          end do
       else
          print *,'unable to distribute $ = ',pooldols(imod),
             ' for mods pool ',modcodes(imod)
       end if
    end do

C   Write out results to a file

    open(80,file='nmod00by_wgt.data')
81  format(i3,i4,i3,8f18.9)

    do imod = begmod, nmod
       do iact = 1, nact
          do iw = 1, nw
             write (80,81) ldcl(imod), iact, iw, varcost(iw,imod,iact),
     &          novarcst(iw,imod,iact), result(iw,imod,iact),
     &          resulta(iw,imod,iact), resultb(iw,imod,iact),
     &          resultf(iw,imod,iact), resultj(iw,imod,iact),work(iw,imod,iact)
          end do
       end do
    end do

    Print *, ' Total Count and Dollars by Matrix '
    write (*,'(2x,a1,i6,f15.2)') 'A', acnt, atot
    write (*,'(2x,a1,i6,f15.2)') 'B', bcnt, btot
    write (*,'(2x,a1,i6,f15.2)') 'C', ccnt, ctot
    write (*,'(2x,a1,i6,f15.2)') 'D', dcnt, dtot
    write (*,'(2x,a1,i6,f15.2)') 'F', fcnt, ftot
    write (*,'(2x,a1,i6,f15.2)') 'G', gcnt, gtot
    write (*,'(2x,a1,i6,f15.2)') 'H', hcnt, htot
    write (*,'(2x,a1,i6,f15.2)') 'J', jcnt, jtot

    print *,'total wgt w/o 6521 = ',wgt6521
    print *,'total wgt inc 6521 = ',wgtall
    nd

C ------------------------------------------------------------------------------------------
c   Assigns shape

    function shapeind(actv,f9635,f9805)

    integer*4 shapeind, actv
    character*1  f9635
    character*4 f9805

    if (((actv.ge.1000).and.(actv.lt.2000)).or.(actv.eq.5431).or.(actv.eq.5441)
     &     .or.(actv.eq.5451).or.(actv.eq.5461)) then
       if (((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K')) then
          shapeind = 1          ! cards
       else
          shapeind = 2          ! letters
       end if
    else if (((actv.ge.2000).and.(actv.lt.3000)).or.(actv.eq.5432).or.(actv.eq.5442)
     &     .or.(actv.eq.5452).or.(actv.eq.5462)) then
       shapeind = 3             ! flats
    else if (((actv.ge.3000).and.(actv.lt.4000)).or.(actv.eq.5433).or.(actv.eq.5443)
     &     .or.(actv.eq.5453).or.(actv.eq.5463)) then
       shapeind = 4             ! IPPs
    else if (((actv.ge.4000).and.(actv.lt.5000)).or.(actv.eq.5434).or.(actv.eq.5444)
     &     .or.(actv.eq.5454).or.(actv.eq.5464)) then
       shapeind = 5             ! parcels
    else
       shapeind = 6             ! other?
    end if

    if (actv.eq.5340) then
       shapeind = 6  !  other
       if (((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K')) then
          shapeind = 1          ! cards
       end if
       if (f9635.eq.'A') then
          shapeind = 2  !  letters
       end if
       if ((f9635.eq.'D').or.(f9635.eq.'E')) then
```

```
         shapeind = 3  !  flats
      end if
      if ((f9635.eq.'F').or.(f9635.eq.'G').or.(f9635.eq.'J')) then
         shapeind = 4     !  IPPs
      end if
      if ((f9635.eq.'H').or.(f9635.eq.'I')) then
         shapeind = 5  !  parcels
      end if
   end if

   if ((actv.ge.10).and.(actv.lt.1000)) then
      if ((f9805(1:1).eq.'1').and.(((f9635.ge.'B').and.(f9635.le.'C')).or.(f9635.eq.'K'))) then
         shapeind = 1  !  cards
      else if (f9805(1:1).eq.'1') then
         shapeind = 2  !  letters
      else if (f9805(1:1).eq.'2') then
         shapeind = 3  !  flats
      else if (f9805(1:1).eq.'3') then
         shapeind = 4  !  IPPs
      else if (f9805(1:1).eq.'4') then
         shapeind = 5  !  parcels
      else
         shapeind = 6  !  other?
      end if
   end if

   return
   end

c ------------------------------------------------------------------------------------------
c     Assigns weight increment

      function weight(f165,if166,if167,ct_nowgt,nw)

      character*1  f165
      integer*4    if166, if167, weight, ct_nowgt, nw

      weight = 0

      if (f165.eq.'A') then
         weight = 1              !  < 1/2 ounce
      else if (f165.eq.'B') then
         weight = 2              !  1 ounces
      else if (f165.eq.'C') then
         weight = 3              !  1 1/2 ounces
      else if (f165.eq.'D') then
         weight = 4              !  2 ounces
      else if (f165.eq.'E') then
         weight = 5              !  2 1/2 ounces
      else if (f165.eq.'F') then
         weight = 6              !  3 ounces
      else if (f165.eq.'G') then
         weight = 7              !  3 1/2 ounces
      else if (f165.eq.'H') then
         weight = 8              !  4 ounces
      else if (f165.eq.'I') then
         if (if166.eq.0) then    !  < 1 lb
            if (if167.gt.0) then
               weight = if167 + 4
            else
               weight = nw
               ct_nowgt = ct_nowgt + 1
            end if
         else if ((if166.eq.1).and.(if167.eq.0)) then
            weight = 20
         else if ((if166.gt.1).or.((if166.eq.1).and.(if167.gt.0))) then
            weight = 21
         else
            weight = nw
            ct_nowgt = ct_nowgt + 1
         end if
      else
         weight = nw
         ct_nowgt = ct_nowgt + 1
      end if

      return
      end
```

```
      program sumclass_nmod_wgt

c     Purpose:  Sum distributed volume-variable mail processing costs for Non-MODs offices to subclass
c               Costs are calculated in the Fortran program nmodproc00_wgt.f

      implicit none

      integer*4 nact, ncl, nmod, nshp, nmat, nshp2, nw

      parameter (nmod = 8)       !  Number of cost pools
      parameter (nact = 255)     !  Number of activity codes
      parameter (ncl = 60)       !  Number of subclasses
      parameter (nshp = 3)       !  Number of shapes
      parameter (nmat = 8)       !  Number of cost categories
      parameter (nshp2 = 5)      !  Number of shapes (class map)
      parameter (nw = 22)        !  Number of weight increments

      real*8    dollars(nmat,nw,nmod,nact)
      real*8   cdols(nmat,nmod,ncl,nshp,nw)

      integer*4 imod, iact, icl, i, j, k, shape, is
      integer*4 ier, shp(nact), iw
      integer*4 clmap(nact), mod(nmod), ldcl(nmod)

      character*14  grp(nmod)
      character*9 class(ncl), clcode
      character*9 class2(ncl)
      character*4 acodes(nact), temp,acin(nshp2)
      character*5 shapetype(nshp)/'1Ltr ','2Flt ','3Pcl '/


      ier = 0

c     Map of cost pools
      open(30,file='costpools.00.nmod.619')
32    format(i4,a14,i5)

      do i = 1, nmod
         read(30,32) mod(i), grp(i), ldcl(i)
      end do
      close(30)
      print *, 'Mod groups read'

c     Map of activity codes
      open(20,file='activity00.ecr.cra')
21    format(a4)

      do i = 1, nact
         read (20,21) acodes(i)
         is = shape(acodes(i))
         shp(i) = is
      end do
      close(20)
      print*, 'Read in activity codes '

      Map of subclasses
      open(33,file='classes_intl.cra.new')
34    format(a9)
      do i = 1, ncl
         read(33,34) class(i)
         class2(i) = class(i)
      end do
      close(33)
      print*, 'Read in classes '

      Maps activity codes to subclass
      open(35,file='classmap_intl.new')
36    format(a9,5(4x,a4))
      do i = 1, nact
         clmap(i) =  0
      end do
      do while (ier.eq.0)
         read(35,36,iostat=ier,end=101) clcode, acin
         do i = 1, nshp2
            j = 0
            if (acin(i).ne.'    ') then
               do iact = 1,nact
                  if (acodes(iact).eq.acin(i)) then
                     j = iact
                  end if
```

```
            end do
            if (j.gt.0) then
               temp = acin(i)
               if (((temp(2:2).eq.'6').or.(temp(2:2).eq.'7').or.
     &             (temp(2:2).eq.'8').or.(temp(1:2).eq.'54'))) then
                  clmap(j) = 17
               else
                  k = 0
                  do icl = 1,ncl
                     if (class2(icl).eq.clcode) then
                        k=icl
                     end if
                  end do
                  if (k.gt.0) then
                     clmap(j) = k
                  else
                     print *,' bad class code = ',clcode,' ',clcode
                  end if
               end if
            else
               print *,' activity code not found ',acin(i)
            end if
         end if
      end do
   end do
101 print *,' read exit of classmap = ',ier
   close(35)

C  Initialize matrices
   do imod = 1, nmod
      do icl = 1, ncl
         do j = 1, nmat
            do is = 1, nshp
               do iw = 1, nw
                  cdols(j,imod,icl,is,iw) = 0.
               end do
            end do
         end do
      end do
   end do

C     .ead in distributed cost data
   open(40,file='nmod00by_wgt.data')
41    format(10x,8f18.9)

   do imod = 1, nmod
      do iact = 1, nact
         do iw = 1, nw
            read (40,41) (dollars(j,iw,imod,iact),j=1,nmat)
         end do
      end do
   end do

C  Sum data to classes

   do j = 1, nmat
      do imod = 1, nmod
         do iact = 1, nact
            do iw = 1, nw
               icl = clmap(iact) !  Subclass for corresponding activity code
               is = shp(iact) !  Assign shape
               if (icl.gt.0) then
                  cdols(j,imod,icl,is,iw) = cdols(j,imod,icl,is,iw)
     &                + dollars(j,iw,imod,iact)
               else
                  print *,' activity ',acodes(iact),' not in class map ', iact
               end if
            end do
         end do
      end do
   end do

   Write out costs by subclass, cost pool, shape, and weight increment
   open(55,file='nmod00_wgt2.csv',recl=500)
6     rmat(a9,',',i2,',',a14,',',i2,',',a5,',',22(f18.9,','))

   do icl = 1, ncl
      do imod = 1, nmod
         do is = 1, nshp
            if ((icl.eq.1).or.(icl.eq.2).or.((icl.ge.8).and.(icl.le.11))) then
```

```
                write(55,56) class(icl), imod, grp(imod), ldcl(imod), shapetype(is),
     &               (cdols(1,imod,icl,is,iw), iw = 1, nw)
             end if
          end do
       end do
    end do

    id


c-------------------------------------------------------------------------------
c     Assign shape

      function shape(act)

      integer*4    shape
      character*4  act

      if (act(1:1).eq.'1') then
         shape = 1  !  Letters
      else if (act(1:1).eq.'2') then
         shape = 2  !  Flats
      else if ((act(1:1).eq.'3').or.(act(1:1).eq.'4')) then
         shape = 3  !  IPPs/Parcels
      else
         shape = 3  !  Other (Special Service)
         if (act.gt.'1000') then
            print*, 'No shape for actv ', act
         end if
      end if

      return
      end
```

Section III: POSTAL SERVICE Method Cost Estimates by Weight Increment– Clerks and Mailhandlers, Window Service

(Programs: admwin_set.f, admwin_wgt2.f, sumclass_wgt.f, win_key_ecr.f)

```
      program admwin_set

c     Purpose:  Prepares the admin/window service IOCS data sets for the R2001-1 cost distribution program
C               through activity code encirclement and conversion of IOCS tally dollar weights
C               to cost pool dollars

        plicit none

      integer*4  numcag,count

      parameter (numcag=11)      !  Number of CAG/finance number combinations

      include 'iocs2000.h'

      integer*4 i,if260,actv,reactv,cagx,searchc,modgrp,ier

      real*8 dollars,rf9250,wgt,totdlr
      real*8 windlr,admdlr
      real*8 bmci,bmcc,nmdi,nmdc
      real*8 win_mod, win_bmc, win_nmod, tot_win

      character*8 costpool
      character*7 iocscag,cagfins(numcag)
      character*5 group


      windlr = 0.0
      admdlr = 0.0
      bmci = 0.0
      bmcc = 0.0
      nmdi = 0.0
      nmdc = 0.0

:     Map of CAG and tally finance numbers
      open(20,file='fincag.98')
      do i=1,numcag
         read(20,'(a7)') cagfins(i)
      end do

      pen(40,file='admwin00.dat',recl=1210) !  Admin/Window Service IOCS output data
1      rmat(a1167,f18.8,i5,a8,1x,a5,i3) ! admwin.dat output format
2     ormat(a1167,f15.5,i2,5x,i5) ! MODS AW format
3     format(a1167,f15.5,7x,i5) ! BMC/Non-MODS AW format

      ier=0
      count=0
      totdlr=0.
      win_mod = 0.0
      win_bmc = 0.0
      win_nmod = 0.0
      tot_win = 0.0

      open(50,file='mods12_aw00by_new.dat',recl=1200) ! MODs 1&2 office admin/window IOCS data
      do while (ier.eq.0)
         read(50,42,iostat=ier,end=60) rec,wgt,modgrp,actv
         if (ier.ne.0) then
            goto 60
         end if
         group='1 mod'
         read(f9250,'(f10.0)') rf9250

      Cost pool assignment
         if (modgrp.eq.95) then
            costpool='2window '
            win_mod = win_mod + rf9250/100000.
            tot_win = tot_win + rf9250/100000.
         else if (modgrp.eq.96) then
            costpool='Intl adm'
         else if (modgrp.eq.99) then
            costpool='2adm out'
         else if (modgrp.eq.97) then
            costpool='2adm inq'
         else
            costpool='2adm    '
         end if

      FinCAG assignment
         iocscag=f263//f264

         cagx = searchc(cagfins,numcag,iocscag)
```

```
      if (cagx.gt.9) then
         cagx = 9
      else if (cagx.eq.0) then
         print*, 'Invalid CAG ', iocscag
      end if                  ! CAGS F-J are combined for distribution purposes

c     JDS activity code encirclement and cost pool dollar conversion
      if (costpool.ne.'2adm out') then
         if (costpool.eq.'2window ') then !  Window Service cost pool
            reactv = 6170
            if ((actv.le.4950).or.((actv.gt.5000).and.(actv.lt.6210)).or.
     &          (actv.eq.6231).or.((actv.ge.6521).and.(actv.le.6523))) then
               reactv = actv
            end if
            if (actv.eq.6330) then
               reactv = 6000
            end if
            if ((actv.ge.5610).and.(actv.le.5700)) then
               reactv=5750
            end if
            dollars=(rf9250/100000.)*765750./866054. !  Window Service cost pool dollar conversion
            count=count+1
            totdlr=totdlr+dollars
            windlr=windlr+dollars
            write(40,41) rec,dollars,reactv,costpool,group,cagx
         else                 !  Administrative cost pools
            reactv=actv
            if (((actv.gt.5000).and.(actv.lt.5200)).or.
     &          ((actv.ge.6000).and.(actv.le.6230)).or.(actv.eq.6240).or.
     &          ((actv.ge.6420).and.(actv.le.6430)).or.
     &          ((actv.ge.6570).and.(actv.le.6580))) then
               reactv=6630
            end if
            if ((actv.eq.5750).and.((f129.eq.'B').or.(f129.eq.'C'))) then
               reactv=6630
            end if
            if ((actv.ge.5610).and.(actv.le.5700)) then
               reactv=5750
            end if
            if (costpool.eq.'2adm inq') then
               dollars=(rf9250/100000.)*20969./15709. !  Claims/Inquiry cost pool dollar conversion
            else if (costpool.eq.'Intl adm ') then
               dollars=(rf9250/100000.)*8395./8965. !  2Adm Intl cost pool dollar conversion
            else
               dollars=(rf9250/100000.)*823585./897210. !  2Adm cost pool dollar conversion
            end if
            count=count+1
            totdlr=totdlr+dollars
            write(40,41) rec,dollars,reactv,costpool,group,cagx
            admdlr=admdlr+dollars
         end if
      end if
   end do

0  print *,'exit of MODS file with ier = ',ier
   print *,'MODS window = ',windlr
   print *,'MODS adm = ',admdlr
   print *,'MODS win tally costs = ', win_mod

   ier=0

   close(50)

   open(50,file='bmcs_aw00by_new.dat',recl=1200) !  BMC admin/window IOCS data
   do while (ier.eq.0)

      read(50,43,iostat=ier,end=65) rec,wgt,actv

      if (ier.ne.0) then
         goto 65
      end if

   group='2 bmc'

      read(f9250,'(f10.0)') rf9250
      read(f260,'(i2)') if260

   Cost pool assignment
      if ((if260.eq.9).or.((if260.ge.24).and.(if260.le.26))) then
```

```
         costpool='2window '
         win_bmc = win_bmc + rf9250/100000.
         tot_win = tot_win + rf9250/100000.
      else if (if260.eq.17) then
         costpool='2adm inq'
      else
         costpool='2adm    '
      end if

c     FinCAG assignment
      iocscag=f263//f264

      cagx = searchc(cagfins,numcag,iocscag)
      if (cagx.gt.9) then
         cagx = 9
      else if (cagx.eq.0) then
         print*, 'Invalid CAG ', iocscag
      end if                 ! CAGS F-J are combined for distribution purposes

c     Activity code encirclement
      reactv=actv
      if ((actv.ge.5610).and.(actv.le.5700)) then
         reactv=5750
      end if

      dollars=(rf9250/100000.)*850132./849454. ! Conversion to BMC cost pool dollars
      count=count+1
      totdlr=totdlr+dollars
      bmci=bmci+rf9250/100000.
      bmcc=bmcc+dollars

      write(40,41) rec,dollars,reactv,costpool,group,cagx

   end do

35 print *,'read exit of BMC file with ier = ',ier
   print *,'BMC IOCS = ',bmci
   print *,'BMC Cost = ',bmcc
   print *,'BMC win tally cost = ', win_bmc

   ier=0

   close(50)

   open(50,file='nonmods_aw00by_new.dat',recl=1200) !  Non-MODs offices admin/window IOCS data
   do while (ier.eq.0)
      read(50,43,iostat=ier,end=70) rec,wgt,actv

      group='3 nmd'

      read(f9250,'(f10.0)') rf9250
      read(f260,'(i2)') if260

   Cost pool assignment
      if ((if260.eq.9).or.((if260.ge.24).and.(if260.le.26))) then
         costpool='2window '
         win_nmod = win_nmod + rf9250/100000.
         tot_win = tot_win + rf9250/100000.
      else if (if260.eq.17) then
         costpool='2adm inq'
      else
         costpool='2adm    '
      end if

   FinCAG assignment
      iocscag=f263//f264

      cagx = searchc(cagfins,numcag,iocscag)
      if (cagx.gt.9) then
         cagx = 9
      else if (cagx.eq.0) then
         print*, 'Invalid CAG ', iocscag
      end if                 ! CAGS F-J are combined for distribution purposes

     tivity code encirclement

      reactv=actv

      if ((actv.ge.5610).and.(actv.le.5700)) then
         reactv=5750
```

```
      end if

      dollars=(rf9250/100000.)*(4833548./4402680.) !  Conversion to Non-MODS cost pool dollars
      count=count+1
      totdlr=totdlr+dollars
      nmdi=nmdi+rf9250/100000.
      nmdc=nmdc+dollars
      write(40,41) rec,dollars,reactv,costpool,group,cagx
   end do

70 print *,'read exit of NONMODS file with ier = ',ier
   print *,'nmd IOCS = ',nmdi
   print *,'nmd Cost = ',nmdc
   print *,'records written= ',count
   print *,'dollars in records= ',totdlr
   print *,'Non-MODS win cost = ', win_nmod
   print *,' '
   print *,'Total window tally costs = ', tot_win

   close(50)
   close(40)

   end
```

```
      program admwin_wgt2

c     Purpose:  Computes distributed volume-variable costs (USPS Method) for Administrative
C               and Window Service cost pools.  Adds additional dimension for weight increments

      implicit none

      integer*4 nmod, nw, nop, ngrp, ncag, nbasic, nitem
      integer*4 nact, ishp, nshp, nact2, nwgt, nwgt2, begmail

C     set appropriate nmod:
      parameter (ngrp = 3)       ! Number of office groups (MODS, Non-MODS, BMC)
      parameter (nop = 24)       ! Number of mail processing IOCS operation codes
      parameter (ncag = 9)       ! Number of fincags
      parameter (nw =  ncag)     ! Number of fincags
      parameter (nmod = 4)       ! Number of adm/win cost pools (2window, 2adm, 2adm inq)
      parameter (nact = 334)     ! Number of direct activity codes
      parameter (nbasic = 4)     ! Number of basic functions
      parameter (nshp = nbasic)  ! Number of basic functions
      parameter (nitem = nbasic) ! Number of basic functions
      parameter (begmail = 15)   ! Set this to the index of the first non-Spec Serv activity code
      parameter (nact2 = 495)    ! Number of total activity codes
      parameter (nwgt = 22)      ! Number of weight increments (including no weight)
      parameter (nwgt2 = 21)     ! Number of weight increments

      include 'iocs2000.h'

      real*8     adols(nw,nwgt,nmod,nact2,nshp) ! Direct costs
      real*8    adist(nw,nwgt,nmod,nact2,nshp) !  Workspace for distributed no weight direct costs
      real*8    cdist(nw,nwgt,nmod,nitem,nact2) ! Workspace for distribution of matrix D
      real*8     cdols(nw,nwgt,nmod,nitem,nact2) ! Other costs
      real*8     ddols(nw,nmod,nitem) ! Mixed mail costs
      real*8    result(nwgt,nmod,nact2) ! Array to hold Results
      real*8    resulta(nw,nwgt,nmod,nact2) ! Array to hold Results for matrix A
      real*8    resultb(nw,nwgt,nmod,nact2) ! Array to hold Results for matrix C
      real*8    actshr(nw,nact2,nwgt)
      real*8     dlrs, sum, distamt,dollars, actwgt(nwgt2)
      real*8    special, overhead
      real*8    atot, btot, ctot, dtot, rf9250

      integer*4 acnt, bcnt, ccnt, dcnt, specialcnt
      integer*4 cnt, iop, igrp
      integer*4 i, j, k, imod, iact, iitem, iw
      integer*4 ier, iwgt, ct_nowgt, if166, if167, weight
      integer*4 searchc, searchi, modgrp, hand, actv
      integer*4 acodes(nact2), class(nact2)
      integer*4 reactv,cagx

      character*8 costpool
      character*5 group
      character*1 codes(26)/'A','B','C','D','E','F','G','H','I','J','K',
     &   'L','M','N','O','P','Q','R','S','T','U','V',
     &   'W','X','Y','Z'/
      character*2 opcodes(nop)/'00','01','02','03','04','05','06','07','08',
     &   '11','12','13','14','15','16','18','19','20','21','22','23',
     $   '27','28','29'/
      character*1 bfcodes(nbasic)/'1','2','3','5'/
      character*8 poolname(nmod)/'2adm    ','2adm inq','2window ','Intl adm'/
      character*5 grpname(ngrp)/'1 mod','2 bmc','3 nmd'/

      special = 0.0
      overhead = 0.0
      atot = 0.0
      btot = 0.0
      ctot = 0.0
      dtot = 0.0
      acnt = 0
      bcnt = 0
      ccnt = 0
      dcnt = 0
      specialcnt = 0
      ier = 0

C     map of activity codes
      open(20,file='activity00.auto.intl2')
21    format(i4,i8)
      do i=1,nact2
         read (20,21) acodes(i), class(i)
      end do
      print *,'read activity map'
```

```
      close(20)

C  Initialize matrices

      do ishp = 1, nshp
         do iact = 1, nact2
            do imod = 1, nmod   ! a matrix
               do iw = 1, nw
                  do iwgt = 1, nwgt
                     adols(iw,iwgt,imod,iact,ishp) = 0.0
                     adist(iw,iwgt,imod,iact,ishp) = 0.0
                  end do
               end do
            end do
         end do
      end do
      do iact = 1, nact2
         do iitem = 1, nitem
            do imod = 1, nmod
               do iw = 1, nw
                  do iwgt = 1, nwgt
                     cdist(iw,iwgt,imod,iitem,iact) = 0.
                     cdols(iw,iwgt,imod,iitem,iact) = 0.
                  end do
               end do
            end do
         end do
      end do
      do iitem = 1, nitem
         do imod = 1, nmod
            do iw=1,nw
               ddols(iw,imod,iitem) = 0.
            end do
         end do
      end do
      do iact = 1, nact2
         do imod = 1, nmod
            do iitem=1,nitem
               do iwgt = 1, nwgt
                  result(iwgt,imod,iact) = 0.
               end do
            end do
         end do
      end do
      do iact = 1, nact2
         do imod = 1, nmod
            do iw = 1, nw
               do iwgt = 1, nwgt2
                  resulta(iw,iwgt,imod,iact) = 0.
                  resultb(iw,iwgt,imod,iact) = 0.
               end do
            end do
         end do
      end do
      print *,'Matrices initialized'

      cnt = 0
      ier = 0
      overhead=0.
      distamt=0.
      ct_nowgt = 0

      open(42,file='admwin00.dat',recl=1200) !  Admin/Window IOCS data

      do while (ier.eq.0)

41       format(a1167,f18.8,i5,a8,1x,a5,i3) ! admwin.dat output format

         read(42,41,iostat=ier,end=100) rec,dollars,reactv,costpool,group,cagx

         overhead=overhead+dollars
         cnt = cnt + 1
         iw = 1

         modgrp = searchc(poolname,nmod,costpool) !  assign cost pool

         if ((modgrp.lt.1).or.(modgrp.gt.nmod)) then
            print *,'bad cost pool = ',costpool,' code = ',modgrp
c            goto 99
         end if
```

```
      read(f262,'(i4)') actv
      read(f9250,'(f10.0)') rf9250
      read(f166,'(i2)') if166
      read(f167,'(i2)') if167

      dlrs = rf9250/100000.

      if ((reactv.ge.5610).and.(reactv.le.5750)) then
         reactv=5750          !  Condense mixed-mail activity codes
      end if

      if ((reactv.ge.10).and.(reactv.le.4950)) then
         hand = 1             ! direct
      else if (reactv.eq.5750) then
         hand = 2             ! mixed mail
      else if (reactv.eq.6522) then
         hand = 3             ! clocking in/out
      else
         hand = 4             ! everything else
      end if

      igrp = searchc(grpname,ngrp,group) !  Assign office type
      ishp = searchc(bfcodes,nbasic,f261) !  Assign basic function
      iitem = ishp
      iact = searchi(acodes,nact2,reactv) !  Assign activity code
      iop = searchc(opcodes,nop,f260) !  Assign operation code

c  Assign weight increment for direct mail tallies only
      if (hand.eq.1) then
         if (reactv.ge.1000) then
            iwgt = weight(f165,if166,if167,ct_nowgt) !  Subroutine to assign weight increment
         else
            iwgt = 22
         end if
      else
         iwgt = 22
      end if

c     Assign PinCAG
      if (cagx.gt.0) then
         iw = cagx
      else
         print *,'bad CAG code = ',cagx
      end if

      if ((hand.eq.1).and.(iact.eq.0)) then
         print *,'missing direct activity code = ',actv,' modgrp = ',modgrp
      end if

c     Direct costs ("A" matrix)
      if (hand.eq.1) then
         if (iact.gt.0) then
            if ((modgrp.gt.0).and.(iw.gt.0)) then
               adols(iw,iwgt,modgrp,iact,ishp)=adols(iw,iwgt,modgrp,iact,ishp) + dollars
               atot = atot + dlrs
               acnt = acnt + 1
               distamt=distamt+dollars
            else
               print *,' bad MODS/CAG in matrix A ', modgrp, iw, dlrs
            end if
         else
            print *,' bad activity in matrix A ',reactv, dlrs
         end if


C*******************************************************************************
c     Mixed-mail costs ("D" matrix)
      else if (hand.eq.2) then
         if ((iw.gt.0).and.(modgrp.gt.0).and.(iitem.gt.0)) then
            ddols(iw,modgrp,iitem) = ddols(iw,modgrp,iitem) + dollars
            distamt=distamt+dollars
            dtot = dtot + dlrs
            dcnt = dcnt + 1
         else
            print *,' bad MODS/CAG in matrix D ', modgrp, costpool,iw, dlrs
         end if

c     Clocking in/out costs ("C" matrix)
      else if (hand.eq.3) then
```

```
         if ((modgrp.gt.0).and.(igrp.gt.0)) then
            cdols(iw,iwgt,modgrp,iitem,iact) = cdols(iw,iwgt,modgrp,iitem,iact) + dollars
            distamt=distamt+dollars
         else
            print *,' bad MODS/CAG in 6522 ', modgrp, costpool,igrp,group
         end if

c     ıl other costs ("C" matrix)
       else if (hand.eq.4) then
          if ((iw.gt.0).and.(modgrp.gt.0).and.(iitem.gt.0).and.(iact.gt.0)) then
             cdols(iw,iwgt,modgrp,iitem,iact) = cdols(iw,iwgt,modgrp,iitem,iact) + dollars
             distamt = distamt+dollars
          else
             print *,' bad MODS/CAG in other ', modgrp, reactv,costpool,iw, dlrs, ' ', f261, ' *'
          end if
       else
          print *,'bad tally classification ',costpool,group,cagx
       end if
99    end do
100   print *,' read exit = ',ier,' with ',cnt,' records '
      print*, 'Total number of direct no weight tallies ', ct_nowgt
      close(42)

C ***********************************End Read Loop*********************************
c  Redistribute direct no weight tallies

      do imod = 1, nmod
         do iitem = 1, nitem
            do iw = 1, nw
               do iact = begmail, nact2
                  if (adols(iw,nwgt,imod,iact,iitem).gt.0.0) then
                     sum = 0.0
                     do iwgt = 1, nwgt2 !  Distribute over all weight increments
                        sum = sum + adols(iw,iwgt,imod,iact,iitem)
                     end do
                     if (sum.gt.0.0) then
                        do iwgt = 1, nwgt2
                           adist(iw,iwgt,imod,iact,iitem) = adist(iw,iwgt,imod,iact,iitem) +
     &                        adols(iw,nwgt,imod,iact,iitem)*adols(iw,iwgt,imod,iact,iitem)/sum
                        end do
                        adols(iw,nwgt,imod,iact,iitem) = 0.0
                     end if
                  end if
               end do
            end do
         end do
      end do

c  Residual distribution of direct no weight tallies
      do imod = 1, nmod
         do iitem = 1, nitem
            do iw = 1, nw
               do iact = begmail, nact2
                  if (adols(iw,nwgt,imod,iact,iitem).gt.0.0) then
                     print*, 'Residual dist of a-matrix no weights ', adols(iw,nwgt,imod,iact,iitem)
                     sum = 0.0
                     do iwgt = 1, nwgt2
                        actwgt(iwgt) = 0.0
                     end do
                     do i = 1, nitem !  Distribute over all basic functions
                        do j = 1, nw !  Distribute over all Fin CAGs
                           do iwgt = 1, nwgt2 !  Distribute over all weight increments
                              actwgt(iwgt) = actwgt(iwgt) + adols(j,iwgt,imod,iact,i)
                              sum = sum + adols(j,iwgt,imod,iact,i)
                           end do
                        end do
                     end do
                     if (sum.gt.0.0) then
                        do iwgt = 1, nwgt2
                           adist(iw,iwgt,imod,iact,iitem) = adist(iw,iwgt,imod,iact,iitem) +
     &                        adols(iw,nwgt,imod,iact,iitem)*(actwgt(iwgt)/sum)
                        end do
                        adols(iw,nwgt,imod,iact,iitem) = 0.0
                     else
                        print*, 'Unable to distribute adols no weights ',
                           adols(iw,nwgt,imod,iact,iitem), ' actv ', acodes(iact),
     &                     ' pool ', poolname(imod)
                     end if
                  end if
               end do
```

```
            end do
         end do
      end do

c     Sum redistributed direct no weight tallies
      do imod = 1, nmod
         do iitem = 1, nitem
            do iw = 1, nw
               do iact = 1, nact2
                  do iwgt = 1, nwgt
                     adols(iw,iwgt,imod,iact,iitem) = adols(iw,iwgt,imod,iact,iitem) +
     &                  adist(iw,iwgt,imod,iact,iitem)
                     adist(iw,iwgt,imod,iact,iitem) = 0.0
                  end do
               end do
            end do
         end do
      end do

c     Distribute mixed-mail costs ("D" matrix) using direct costs ("A" matrix) as distribution key
C     over all basic functions and DBMC categories

      do iitem = 1, nitem       !  Basic function
         if (iitem.ne.4) then   ! BF 5 treated differently
            do imod = 1, nmod   !  Cost pool
               do iw = 1,nw     !  Fin CAG
                  if (ddols(iw,imod,iitem).gt.0.) then
                     sum = 0.
                     do iact = 1, nact !  Distribute over all direct activity codes
                        do iwgt = 1, nwgt !  Distribute over all weight increments
                           sum = sum + adols(iw,iwgt,imod,iact,iitem)
                        end do
                     end do
                     if (sum.gt.0) then
                        do iact = 1, nact
                           do iwgt = 1, nwgt
                              cdist(iw,iwgt,imod,iitem,iact) = cdist(iw,iwgt,imod,iitem,iact) +
     &                           ddols(iw,imod,iitem) * adols(iw,iwgt,imod,iact,iitem) / sum
                           end do
                        end do
                        ddols(iw,imod,iitem) = 0.
                     end if
                  end if
               end do
            end do
         end if
      end do

c     Distribute mixed-mail costs ("D" matrix) using direct costs ("A" matrix) as distribution key
c     over all basic functions and DBMC categories

      do iitem = 1, nitem       !  Basic function
         do imod = 1, nmod      !  Cost pool
            do iw = 1,nw        !  Fin CAG
               if (ddols(iw,imod,iitem).gt.0.) then
                  print *,'residual distribution of mm in pool = ',imod,' ',iitem,
     &                 ' ',iw,', ',ddols(iw,imod,iitem)
                  sum = 0
                  do iact = 1, nact
                     do iwgt = 1, nwgt
                        actshr(1,iact,iwgt) = 0.
                     end do
                  end do
                  do iact = 1, nact !  Distribute over all direct activity codes
                     do j = 1, nitem !  Distribute over all basic functions
                        do k = 1, nwgt !  Distribute over all weight increments
                           actshr(1,iact,k) = actshr(1,iact,k) + adols(iw,k,imod,iact,j)
                           sum = sum + adols(iw,k,imod,iact,j)
                        end do
                     end do
                  end do
                  if (sum.gt.0.) then
                     do iact = 1, nact
                        do iwgt = 1, nwgt
                           cdist(iw,iwgt,imod,iitem,iact) = cdist(iw,iwgt,imod,iitem,iact) +
                              ddols(iw,imod,iitem) * actshr(1,iact,iwgt) / sum
                        end do
                     end do
                  else
                     print *,' unable to dist D dols for pool = ',imod,' ',iitem,
```

```
     &                ' ',iw,', ',ddols(iw,imod,iitem)
c       Dump mixed mail costs not distributed in activity code 5750  ****
                  cdols(iw,nwgt,imod,iitem,384) = cdols(iw,nwgt,imod,iitem,384) + ddols(iw,imod,iitem)
               end if
            end if
         end do
      end do
     nd do

c     Sum distributed mixed-mail costs into all other costs ("C" matrix)
      do iact = 1, nact2
         do iitem = 1, nitem
            do imod = 1, nmod
               do iw = 1, nw
                  do iwgt = 1, nwgt
                     cdols(iw,iwgt,imod,iitem,iact) = cdols(iw,iwgt,imod,iitem,iact) + cdist(iw,iwgt,imod,iitem,iact)
                     cdist(iw,iwgt,imod,iitem,iact) = 0.
                  end do
               end do
            end do
         end do
      end do

c     Sum all costs into a single array (weight increment,cost pool,activity code)
c     Direct costs
      do ishp = 1, nshp
         do iact = 1, nact2
            do imod = 1, nmod
               do iw = 1, nw
                  do iwgt = 1, nwgt
                     result(iwgt,imod,iact) = result(iwgt,imod,iact) + adols(iw,iwgt,imod,iact,ishp)
                  end do
               end do
            end do
         end do
      end do
c     Indirect costs
      do iact = 1, nact2
         do iitem = 1, nitem
            do imod = 1, nmod
               do iw = 1, nw
                  do iwgt = 1, nwgt
                     result(iwgt,imod,iact) = result(iwgt,imod,iact) + cdols(iw,iwgt,imod,iitem,iact)
                  end do
               end do
            end do
         end do
      end do

c     Write out window service results
      open(80,file='awdist00_wgt.data')
81    format(i2,1x,a8,i4,i5,i3,1x,i3,1x,f18.9)

      do imod = 3, 3
         do iact = 1, nact2
            do iwgt = 1, nwgt
               if (result(iwgt,imod,iact).gt.0.) then
                  write (80,81) imod,poolname(imod), iact, acodes(iact), class(iact), iwgt,
     &               result(iwgt,imod,iact)
               end if
            end do
         end do
      end do

      Print *, ' Total Count and Dollars by Matrix '
      write (*,'(2x,a1,i6,f15.2)') 'A', acnt, atot
      write (*,'(2x,a1,i6,f15.2)') 'B', bcnt, btot
      write (*,'(2x,a1,i6,f15.2)') 'C', ccnt, ctot
      write (*,'(2x,a1,i6,f15.2)') 'D', dcnt, dtot
      write (*,'(2x,a1,i6,f15.2)') 'S', specialcnt, special

      print *,'total dollars read = ',overhead
      print *,'total dollars in = ',distamt

     id

c ---------------------------------------------------------------------------------------
c     Assign weight increment

      function weight(f165,if166,if167,ct_nowgt)
```

```
character*1  f165
integer*4    if166, if167, weight, ct_nowgt

weight = 0

 f (f165.eq.'A') then
   weight = 1              !  < 1/2 ounce
else if (f165.eq.'B') then
   weight = 2              !  1 ounces
else if (f165.eq.'C') then
   weight = 3              !  1 1/2 ounces
else if (f165.eq.'D') then
   weight = 4              !  2 ounces
else if (f165.eq.'E') then
   weight = 5              !  2 1/2 ounces
else if (f165.eq.'F') then
   weight = 6              !  3 ounces
else if (f165.eq.'G') then
   weight = 7              !  3 1/2 ounces
else if (f165.eq.'H') then
   weight = 8              !  4 ounces
else if (f165.eq.'I') then
   if (if166.eq.0) then    !  < 1 lb
      if (if167.gt.0) then
         weight = if167 + 4
      else
         weight = 22
         ct_nowgt = ct_nowgt + 1
      end if
   else if ((if166.eq.1).and.(if167.eq.0)) then
      weight = 20
   else if ((if166.gt.1).or.((if166.eq.1).and.(if167.gt.0))) then
      weight = 21
   else
      weight = 22
      ct_nowgt = ct_nowgt + 1
   end if
else
   weight = 22
   ct_nowgt = ct_nowgt + 1
 nd if

return
end
```

```
      program sumclass_wgt

c     Purpose:  Summarizes the distributed window service costs to subclass

      implicit none

      integer*4 nact, ncl, nmod, nshp, nwgt, ncl2

      parameter (nmod = 3)       !  Number of cost pools
      parameter (nact = 495)     !  Number of activity codes
      parameter (ncl = 212)      !  Number of subclasses
      parameter (ncl2 = 73)      !  Number of CRA subclasses
      parameter (nshp = 4)       !  Number of shapes
      parameter (nwgt=22)        !  Number of weight increments

      real*8    cost_aw(nmod,ncl,nshp,nwgt)
      real*8    cost

      integer*4 imod, iact, icl, i, searchc, shape, is
      integer*4 ier, shp(nact), iw, iwgt, ct, ishp

      character*9 class(ncl)
      character*4 acodes(nact), actv
      character*12 wgtinc(nwgt)
      character*7 group(nmod) /'adm    ','adm inq','window '/
      character*5 shapetype(nshp) /'1Ltr ','2Flt ','3Pcl ','4None'/

c     Map of activity codes
      open(10,file='activity00.auto.intl2')
14    format(a4)
      do i = 1, nact
         read (10,14) acodes(i)
         is = shape(acodes(i))
         shp(i) = is
      end do
      print *,'read activity map'
      close(10)

c     Map of subclasses
      open(10,file='classes.auto.intl2')
11    format(a9)
      do i = 1, ncl
         read(10,11) class(i)
      end do
      close(10)
      print*, 'Classes read in '

c     Map of weight categories
      open(10,file='wgtinc.prn2')
12    format(a12)
      do i = 1, nwgt
         read(10,12) wgtinc(i)
      end do
      close(10)
      print*, 'Weight read in '

c     Initialize matrices
      do imod = 1, nmod
         do icl = 1, ncl
            do ishp = 1, nshp
               do iwgt = 1, nwgt
                  cost_aw(imod,icl,ishp,iwgt) = 0.0
               end do
            end do
         end do
      end do
      print*, 'Matrices initialized '

c     Read in distributed admin/window service costs from admwin_wgt2.f
      open(40,file='awdist00_wgt.data')
41    format(i2,9x,5x,a4,i3,1x,i3,1x,f18.9)

      ier = 0
      ct = 0

      do while (ier.eq.0)

         read (40,41,iostat=ier,end=100) imod, actv, icl, iwgt, cost

         ct = ct + 1
```

```
      if (actv(1:1).eq.' ') actv(1:1)='0'
      if (actv(2:2).eq.' ') actv(2:2)='0'

      iact = searchc(acodes,nact,actv) !  Find activity code

      if (iact.gt.0) then

         is = shp(iact)      !  Assign shape

         if ((icl.ge.31).and.(icl.le.63)) icl = 30 !  Condense Intl
         if ((iact.ge.356).and.(iact.le.377)) icl = 30 !  Condense Intl
         if (icl.eq.112) icl = 68 !  Money Order
         if ((icl.eq.69).or.(icl.eq.84).or.(icl.eq.109).or.(icl.eq.114)) icl = 70 !  Stamp Env (incl Stamped Cards)

         cost_aw(imod,icl,is,iwgt) = cost_aw(imod,icl,is,iwgt) + cost

      end if

   end do

100   print*, 'Read exit error ', ier, ' record ct = ', ct

c     Write out window service costs by subclass, shape, and weight increment
      open(50,file='wincost_wgt00.csv')
51    format(i2,',',a7,',',i2,',',a9,',',a5,',',i2,',',a12,',',f15.5)
      do imod = 3, nmod
         do icl = 1, ncl2
            do ishp = 1, nshp
               do iw = 1, nwgt
                  if ((icl.lt.31).or.(icl.gt.63)) then
                     write (50,51) imod, group(imod), icl, class(icl), shapetype(ishp),
     &                    iw, wgtinc(iw), cost_aw(imod,icl,ishp,iw)
                  end if
               end do
            end do
         end do
      end do

      end


c-----------------------------------------------------------------------------
c     Assign shape

      function shape(act)

      integer*4    shape
      character*4  act

      if (act(1:1).eq.'1') then
         shape = 1  !  Letters
      else if (act(1:1).eq.'2') then
         shape = 2  !  Flats
      else if ((act(1:1).eq.'3').or.(act(1:1).eq.'4')) then
         shape = 3  !  IPPs/Parcels
      else
         shape = 4  !  Other
      end if

      return
      end
```

```
      program win_key_ecr

c     Purpose:  To create the window service dist key for Fcn 4 support cost pool distribution
C               by actv code, weight category, and ECR category
C               using BY00 CRA Cost Segment 3.2 window service costs

      implicit none

      integer*4  nact, ncat

      parameter (nact=255)       !  Number of direct actv Codes
      parameter (ncat=22)        !  Number of weight increments

      integer*4  iact, icat, searchc, ier, ct, cat

      real*8     tot_dols, dlrs, cost(nact,ncat)
      real*8     cost_mp(nact)

      character*4  acodes(nact), actv


c     Map of activity codes
      open(10,file='activity00.ecr.cra')
11    format(a4)

      do iact = 1, nact
         read(10,11) acodes(iact)
      end do
      print*, 'Actv codes read in '
      close(10)

c     Initialize matrices
      do iact = 1, nact
         do icat = 1, ncat
            cost(iact,icat) = 0.0
            cost_mp(iact) = 0.0
         end do
      end do
      print*, 'Matrices initialized '

      ier = 0
      ct = 0
      tot_dols = 0.0

c     CRA C/S 3.2 costs - Workbook 'win cost by oz 00 new.xls', worksheet "ECR Actv"
      open(20,file='windkecr00.prn')
21    format(a4,i7,f15.5)

      do while (ier.eq.0)

         read(20,21,iostat=ier,end=100) actv, cat, dlrs

         ct = ct + 1
         tot_dols = tot_dols + dlrs

         iact = searchc(acodes,nact,actv) !  activity code

         if (iact.gt.0) then
            cost(iact,cat) = cost(iact,cat) + dlrs
            cost_mp(iact) = cost_mp(iact) + dlrs
         else
            print*, 'Actv code not found ', actv
         end if

      end do
100   print*, 'Read exit error ', ier
      print*, 'Total records ', ct, ' cost ', tot_dols

c     Writes out Function 4 Support distribution key map - used in the FORTRAN program modsproc00_wgt.f
      open(30,file='windk_wgt_ecr.00.619')
31    format(a4,1x,i2,1x,f15.5)

      do iact = 1, nact
         do icat = 1, ncat
            write(30,31) acodes(iact), icat, cost(iact,icat)
         end do
      end do

      end
```

Section IV: City Carrier In-Office Costs by Weight Increment

(Programs: encode_wgt.f, encdata.sm, liocatt.f, fillmixmap.f, loaddata.f, fungroup.f, sortcost.f, level1.f, level2.f, sortlev2a.f, level3.f, report.f, rpt_wgt22cra.f)

```
      PROGRAM encode_wgt
C
C     Purpose: Encode tallies with indexes of arrays instead of
C              actual data. Delete leave tallies. Split old group 26
C              into new sub groups, change nonmod groups
C
C       dified:  For weight increment analysis for R2001-1
c                Also performs the ECR breakout for delivery study

      IMPLICIT NONE

      integer*4   numcf, numact, numor, nw

      parameter   (numcf = 11)  ! number of cag - finnance number combs.
      parameter   (numor = 17)  ! number of operation/route codes
      parameter   (numact = 501) ! number of activity codes
      parameter   (nw = 22)     ! number of weight increments

      include 'iocs2000.h'

      integer*4   desigind      ! function to assign pay category
      integer*4   orind         ! function to assign operation/route code
      integer*4   bfind         ! function to assign basic function index
      integer*4   weight        ! function to assign weight increment
      integer*4   i2, i3, i4, i5, i6, i8, i7, i9
      integer*4   i, searchc, z, totall, ier, countlv
      integer*4   counto, countg, countsp, countk, countf
      integer*4   if166, if167, ct_nowgt

      real*8    tvalue, cost_clk, cost_mp

      character*7    cagfins(numcf), fincag
      character*4    acodes(numact), activity

      countlv = 0
      counto = 0
      countg = 0
      countsp = 0
      countk = 0
      countf = 0
      totall = 0
      ier = 0
      i2 = 0
      i3 = 0
      i4 = 0
      i5 = 0
      i6 = 0
      i7 = 0
      i8 = 0
      i9 = 0

c     Map of Fin CAG combinations (used for strata)
      open(14,file='fincag.98',iostat=ier)
      if (ier.ne.0) then
         print *,' error opening cagfin.s = ',ier
         stop
      end if

15    format(a7)
      do i = 1,numcf
         read(14,15) cagfins(i)
      end do
      print *,'finished reading cags  '

c     Map of activity codes
      open(16,file='activity00.ecr.all',iostat=ier )
      if (ier.ne.0) then
         print *,' error opening activity code files = ',ier
         stop
      end if

17    format(a4)
      do i = 1,numact
         read(16,17) acodes(i)
      end do
      close (14)
      close (16)
      print *,'finished reading activity codes  '

      open(25,file='iocsdata.2000.new',recl=1200) !  FY00 IOCS Data Set (based on full data set)
```

```
      open(30,file='encdata')
21    format(a)
31    format(i2,i1,i1,i3,i2,i3,i2,f11.2)

      ier = 0
      z=0
      st_clk = 0.0
      st_mp = 0.0
      ct_nowgt = 0

      do while (ier.eq.0)
         read(25,21,iostat=ier,end=100) rec

         z=z+1
         fincag = f263//f264     !  Strata

         i2 = searchc(cagfins,numcf,fincag) !  Find fincag (strata)
         i3 = desigind(f257)     ! craft
         i4 = orind(f260)        ! operation or route code
         if (i3.eq.1) i4 = 1     ! supervisors have no route or oper code
         i5 = bfind(f261)        ! basic function
         if (i3.eq.1) i5 = 1     ! supervisors have bf set to 0

         activity = f9806        ! activity code

         if (activity(1:1).ge.'1'.and.activity(1:1).le.'4'.and. ! map x091 to x080
     &      activity(2:4).eq.'091') then
            activity(2:4) = '080'
         end if

         if (i3.eq.2) then
            if (i4.eq.13) i4 = 11 !  Dump Op 88 into other for clk/mh
         end if

c     ECR activity code assignment
         if (activity(2:4).eq.'060') then !  1st Single Piece
            if (f136.eq.'D') then !  Metered
               if (activity.eq.'1060') activity = '1068'
               if (activity.eq.'2060') activity = '2068'
               if (activity.eq.'3060') activity = '3068'
               if (activity.eq.'4060') activity = '4068'
            end if
         end if
         if (activity(2:4).eq.'310') then !  Std A ECR
            if (f9618.eq.'1') then !  WSH
               if (activity.eq.'1310') activity = '1311'
               if (activity.eq.'2310') activity = '2311'
               if (activity.eq.'3310') activity = '3311'
               if (activity.eq.'4310') activity = '4311'
            else if (f9619.eq.'1') then !  WSS
               if (activity.eq.'1310') activity = '1313'
               if (activity.eq.'2310') activity = '2313'
               if (activity.eq.'3310') activity = '3313'
               if (activity.eq.'4310') activity = '4313'
            else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
               if (activity.eq.'1310') activity = '1312'
               if (activity.eq.'2310') activity = '2310'
               if (activity.eq.'3310') activity = '3310'
               if (activity.eq.'4310') activity = '4310'
            else                    !  ECRLOT
               activity = activity
            end if
         end if
         if (activity(2:4).eq.'330') then !  Std A Nonprofit ECR
            if (f9618.eq.'1') then !  WSH
               if (activity.eq.'1330') activity = '1331'
               if (activity.eq.'2330') activity = '2331'
               if (activity.eq.'3330') activity = '3331'
               if (activity.eq.'4330') activity = '4331'
            else if (f9619.eq.'1') then !  WSS
               if (activity.eq.'1330') activity = '1333'
               if (activity.eq.'2330') activity = '2333'
               if (activity.eq.'3330') activity = '3333'
               if (activity.eq.'4330') activity = '4333'
            else if ((f9612.eq.'1').or.(f9613.eq.'1').or.(f9614.eq.'1')) then !  AutoECR
               if (activity.eq.'1330') activity = '1332'
               if (activity.eq.'2330') activity = '2330'
               if (activity.eq.'3330') activity = '3330'
               if (activity.eq.'4330') activity = '4330'
            else                    !  ECRLOT
```

```
            activity = activity
          end if
        end if

        i6 = searchc(acodes,numact,activity)

        if (i6.eq.0) then
          print*, 'Activity code not found ', activity
        end if

        i9 = 1

        read(f166,'(i2)') if166
        read(f167,'(i2)') if167

c     Weight increment assignment
        if (((activity.ge.'1000').and.(activity.le.'4950')).or.
     &      ((activity.ge.'5300').and.(activity.le.'5480'))) then
          i7 = weight(f165,if166,if167,ct_nowgt,nw) ! weight increments for direct actv codes only
        else
          i7 = nw
        end if

        read(f9250,'(f10.2)') tvalue !  F9250

        if ((i2.gt.0).and.(i3.gt.0).and.
     +      (i4.gt.0).and.(i5.gt.0).and.(i6.gt.0).and.(i7.gt.0)) then
          write (30,31) i2, i3, i5, i6, i9, i7, i4, tvalue
          countg = countg + 1
          if (i3.eq.2) then   ! clk/mh
            cost_clk = cost_clk + tvalue
            if ((i4.le.12).and.(i4.ne.10).and.(i4.ne.11)) then
              cost_mp = cost_mp + tvalue
            end if
          end if
        else
          if (f9806(1:1).eq.'9') then ! first digit of activity code f262
            countlv = countlv + 1
          else if (f257(2:2).eq.'4') then ! second digit of da code f257
            countsp = countsp + 1
          else if (f264.eq.'K') then ! cag k tally  f264
            countk = countk + 1
          else if (i2.eq.0) then
            print *,' invalid cagfin = ', fincag, ' fin = ', f2,
     &        ' roster ', f257, ' op code ', f260
            countf = countf + 1
          else
            print *,'    indexes = ',i2,i3,i4,i5,i6,i8
            print*, ' f260 = ', f260, ' f261 = ', f261
            counto = counto + 1
          end if
        end if
      end do
100   print *,' read exit code = ',ier
      print *,' number of records written to encdata = ',countg
      print *,' number of leave records excluded    = ',countlv
      print *,' number of spec. delivery excluded   = ',countsp
      print *,' number of CAG K records excluded    = ',countk
      print *,' number of invalid finance numbers   = ',countf
      print *,' number of other records excluded    = ',counto

      print*, 'Total valid clk/mh costs = ', cost_clk
      print*, 'Total valid clk/mh mail proc costs = ', cost_mp

      end


C ------------------------------------------------------------------------------
C       desigind
C
C       assigns index of 1-6 based on roster designation

        function    desigind(char)

        integer*4       desigind
        character*2     char

        if ((char.eq.' 9').or.
     +      (char.eq.'09').or.
     +      (char.eq.'19')) then
```

```
         desigind = 1
      else if (char.eq.'11') then
         desigind = 2
      else if ((char.eq.'31').or.
     +         (char.eq.'41').or.
     +         (char.eq.'61').or.
     +         (char.eq.'81')) then
         desigind = 2
      else if ((char.eq.'12').or.
     +         (char.eq.'32').or.
     +         (char.eq.'42').or.
     +         (char.eq.'62').or.
     &         (char.eq.'82')) then
         desigind = 2
      else if (char.eq.'13') then
         desigind = 3
      else if ((char.eq.'33').or.
     +         (char.eq.'43').or.
     +         (char.eq.'63').or.
     &         (char.eq.'83')) then
         desigind = 3
      else
         desigind = -1
      end if

      return
      end
C -------------------------------------------------------------------------
C    bfind
C
C    returns index for basic function

      function    bfind(char)

      integer*4      bfind
      character*1    char

      if (char.eq.'1') then
         bfind = 1
      else if (char.eq.'2') then
         bfind = 2
      else if (char.eq.'3') then
         bfind = 3
      else if (char.eq.'5') then
         bfind = 4
      else
         bfind = -1
      end if

      return
      end
C -------------------------------------------------------------------------
C     orind

c     returns index value for operation or route code

      function    orind(char)

      integer*4      orind
      character*2    char

      if (char.eq.'00') then
         orind = 1
      else if (char.eq.'01') then
         orind = 2
      else if (char.eq.'02') then
         orind = 3
      else if (char.eq.'03') then
         orind = 4
      else if (char.eq.'04') then
         orind = 5
      else if ((char.eq.'05').or.
     +         ((char.ge.'11').and.(char.le.'13')).or.
     +         (char.eq.'15').or.
     +         (char.eq.'16').or.
     +         ((char.ge.'19').and.(char.le.'21')).or.
     +         ((char.ge.'27').and.(char.le.'29'))) then
         orind = 6
      else if ((char.eq.'06').or.
     +         (char.eq.'18').or.
```

```
     +           (char.eq.'22').or.
     +           (char.eq.'23')) then
            orind = 7
         else if (char.eq.'07') then
            orind = 8
         else if (char.eq.'08') then
            orind = 9
         else if ((char.eq.'09').or.
     +           (char.eq.'24').or.
     +           (char.eq.'25').or.
     +           (char.eq.'26')) then
            orind = 10
         else if ((char.eq.'10').or.
     +           (char.eq.'17')) then
            orind = 11
         else if (char.eq.'14') then
            orind = 12
         else if (char.eq.'71') then
            orind = 1
         else if (char.eq.'73') then
            orind = 2
         else if (char.eq.'75') then
            orind = 3
         else if (char.eq.'77') then
            orind = 4
         else if (char.eq.'78') then
            orind = 5
         else if (char.eq.'80') then
            orind = 6
         else if (char.eq.'82') then
            orind = 7
         else if (char.eq.'83') then
            orind = 8
         else if (char.eq.'84') then
            orind = 9
         else if (char.eq.'85') then
            orind = 10
         else if (char.eq.'86') then
            orind = 11
         else if (char.eq.'87') then
            orind = 12
         else if (char.eq.'88') then
            orind = 13
         else if (char.eq.'89') then
            orind = 14
         else if (char.eq.'90') then
            orind = 15
         else if (char.eq.'98') then
            orind = 16
         else if (char.eq.'99') then
            orind = 17
         else
            orind = -1
         end if

         return
         end

C ------------------------------------------------------------------------------------------
C     weight

C     Assigns weight increment

      function weight(f165,if166,if167,ct_nowgt,nw)

      character*1  f165
      integer*4    if166, if167, weight, ct_nowgt, nw

      weight = 0

      if (f165.eq.'A') then
         weight = 1             !  < 1/2 ounce
      else if (f165.eq.'B') then
         weight = 2             !  1 ounces
      else if (f165.eq.'C') then
         weight = 3             !  1 1/2 ounces
      else if (f165.eq.'D') then
         weight = 4             !  2 ounces
      else if (f165.eq.'E') then
         weight = 5             !  2 1/2 ounces
```

```
else if (f165.eq.'F') then
   weight = 6              !  3 ounces
else if (f165.eq.'G') then
   weight = 7              !  3 1/2 ounces
else if (f165.eq.'H') then
   weight = 8              !  4 ounces
 'se if (f165.eq.'I') then
   if (if166.eq.0) then    !  < 1 lb
      if (if167.gt.0) then
         weight = if167 + 4
      else
         weight = nw
         ct_nowgt = ct_nowgt + 1
      end if
   else if ((if166.eq.1).and.(if167.eq.0)) then
      weight = 20
   else if ((if166.gt.1).or.((if166.eq.1).and.(if167.gt.0))) then
      weight = 21
   else
      weight = nw
      ct_nowgt = ct_nowgt + 1
   end if
else
   weight = nw
   ct_nowgt = ct_nowgt + 1
end if

return
end
```

```
% Name: Encdata.sm
% Sort encoded IOCS tally info
input file is 'encdata', recs are data sensitive upto 30 chars.
output file is 'encdata.s', recs are data sensitive  upto 30 chars.
sort.
end
```

```
      PROGRAM liocatt

C     PURPOSE: Allocate costs to raw tallies, and performs LIOCATT mixed mail cost distribution

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   nlev1a, nlev1b, nfun
      integer*4   nlev2a, nlev2b, nlev3a, nlev3b

      character*3 runtype

      logical     dofun

      call  getarg(1,runtype)
      if (runtype.eq.'fun') then
         dofun = .true.
      else
         dofun = .false.
      end if

      call  fillmixmap          ! load mixed mail distribution map and activity codes
      call  loaddata            ! load encdata.s (from encode_wgt.f, encdata.sm)
      call  fungroup(nfun)      ! form function groups from operations
      call  sortcost(nfun)      ! sort records for level1
      call  level1(nfun,nlev1a,nlev1b) ! level 1 indirect cost allocation
      call  level2(nlev1a,nlev2a,nlev2b) ! level 2 indirect cost allocation
      call  sortlev2a(nlev2a)   ! sort records for level 3
      call  level3(nlev2a,nlev3a,nlev3b) ! level 3 indirect cost allocation
      call  report(nlev1b,nlev2b,nlev3a,nlev3b) ! write results to file
      print *,' ************************************************** '
      print *,' Fiscal year 2000 completed '
      print *,' Total number of records = ', nrec
      print *,' Number of records after function creation = ',nfun
      print *,' Number of level 1 records = ',nlev1a,', ',nlev1b
      print *,' Number of level 2 records = ',nlev2a,', ',nlev2b
      print *,' Number of level 3 records = ',nlev3a,', ',nlev3b
      print *,' ************************************************** '

      end

C -------------------------------------------------------------------------------
```

```
      subroutine fillmixmap

C     PURPOSE: Read the mixed mail codes and produce map for distributing
C              mixed mail codes to appropriate direct activity codes

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   i, j, ind
      integer*4   ier
      integer*4   searchc

      character*4 mmcodes(nummix)
      character*4 codes(20)

      logical  flag

      if (debug) print *,' Enter subroutine fillmixmap '

      ier = 0

c     Map of activity codes
      open(16,file='activity00.ecr.all',iostat=ier )
17    format(a4)
      do i = 1,numact
         read(16,17) acodes(i)
      end do

C       initialize count array (number of direct keys indirect code is distributed accross)
      do i =1 , nummix
         count(i) = 0
      end do

c     Map of mixed mail activity codes
      open(18,file='mmcodes.intl')
19    format(a4)
      do i = 1,nummix
         read(18,19) mmcodes(i)
      end do

c     Maps mixed mail codes to direct activity codes
      open(20,file='mxmail.all.ecr')
21    format(20a4)

      do while (ier.eq.0)
         read (20,21,iostat=ier,end=100) codes
         i = searchc(mmcodes,nummix,codes(1))
         if (i.gt.0) then
            flag = .true.
            ind = 1
            do while ((flag).and.(ind.lt.20))
               ind = ind + 1
               if (codes(ind).ne.'    0') then
                  j = searchc(acodes,numact,codes(ind))
                  if (j.gt.0) then
                     count(i) = count(i) + 1
                     mixmap(count(i),i) = j
                  else
                     print *,' Direct mail code did not map ',codes(ind)
                  end if
               else
                  flag = .false.
               end if
            end do
         else
            print *,' Mixed mail code did not map ',codes(1)
         end if
      end do
100   print *,' read exit code = ',ier

C     Fill mix_to_act array

      do i = 1,nummix
         mix_to_act(i) = searchc(acodes,numact,mmcodes(i))
      end do

      if (debug) print *,' exiting fillmixmap '

      close (16)
```

```
close (18)
close (20)

return
end
```

```
      subroutine loaddata

c     Purpose:  To read in coded data set - encode_wgt.f, encdata.sm

      IMPLICIT NONE

       include 'liocatt.h'

      integer*4   j
      integer*2   ioff, iact, ibf, iw, ifun, ipig, iocc
      integer*4   ier/0/
      character*14   datum, ldatum
      real*8         cost, lcost, tcost


      open (20,file='encdata.s') !  Sorted data set
21    format(i2,i1,i1,i3,i2,i3,i2,f11.2)

      lcost = 0.0
      tcost = 0.0
      ldatum = ' '
      j = 0
      nrec = 0
      do while (ier.eq.0)
         read (20,21,iostat=ier,end=100) ioff,iocc,ibf,iact,ipig,iw,ifun, cost
         write (datum,'(7a2)') ioff,iocc,ibf,iact,iw,ipig,ifun
         if ((datum.ne.ldatum).and.(j.ne.0)) then
            nrec = nrec + 1
            if (nrec.le.maxcost) then
               write (costbuf2(nrec),'(a14,a8)') ldatum, tcost
            else
               print *,' maxcost exceeded in loaddata '
               stop
            end if
            ldatum = datum
            tcost = cost
         else
            if (j.eq.0) then
               j = 1
               ldatum = datum
            end if
            tcost = tcost + cost
         end if
      end do
100   if (debug) print *,' Read exit of encdata = ',ier,', nrec = ',nrec
      close (20)
      return
      end
```

```
      subroutine fungroup(numout)

C     PURPOSE: Add clerk/mail handler records up into functional groups from operation codes.

      IMPLICIT NONE

      include  'liocatt.h'

      real*8       original_costs(numopr)
      real*8       fun_costs(numfun)
      real*8       cost

      integer*2    opr_to_fun(numfun,numopr)
      integer*2    cofg, cpay, copr, cbf, cact, cw, cpig
      integer*2    ofg, pay, opr, fun, bf, act, w, pig
      integer*4    i, j, numout
      integer*4    ier/0/
      integer*4    indb1, indb2

      logical      same

      if (debug) print *,' entering fungroup.f77 '

C     Fill opr_to_fun array

      open(20,file='operrtemap')
21    format(45i3)
      do i = 1,numfun
         read (20,21) (opr_to_fun(i,j),j=1,numopr)
      end do

C     open input and output file

C     Collect data for a office, pay category cell.
C     1.  Collapse over quarter
C     2.  If pay category is clerk/mailhandler create functional groups
C     3.  Output for regular processing in level1 - level3

31    format(7a2,a8)

C       ead first record of first group

      same = .true.
      indb1 = 1
      indb2 = 0
      do opr = 1, numopr
         original_costs(opr) = 0.0
      end do

      read (costbuf2(indb1),31) cofg, cpay, cbf, cact, cw, cpig, copr, cost
      original_costs(copr) = original_costs(copr) + cost

      do while (ier.eq.0)

C     Read rest of cost group
         do while (same)
            indb1 = indb1 + 1
            if (indb1.gt.nrec) then
               ier = -1
               goto 100
            end if
            read (costbuf2(indb1),31) ofg, pay, bf, act, w, pig, opr, cost
            if ((ofg.eq.cofg).and.(pay.eq.cpay).and.(bf.eq.cbf).and.
     +         (act.eq.cact).and.(w.eq.cw).and.(pig.eq.cpig)) then
               original_costs(opr) = original_costs(opr) + cost
            else
               copr = opr
               same = .false.
            end if
         end do
100      if (ier.ne.0) print *,' Read exit code = ',ier

C     Sum operation codes into function groups if clerk/mailhandler

         if (cpay.eq.2) then
            do fun = 1,numfun
               fun_costs(fun) = 0.
               do i = 2,opr_to_fun(fun,1)
                  fun_costs(fun) = fun_costs(fun) +
     +               original_costs(opr_to_fun(fun,i))
```

```
            end do
         end do
      end if

C     Output data from original costs if supervisor or carrier,
C     and from fun costs if clerk/mailhandler

      if ((cpay.eq.1).or.(cpay.eq.3)) then
         do opr = 1, numopr
            if (original_costs(opr).gt.0) then
               indb2 = indb2 + 1
               if (indb2.gt.maxcost) then
                  print *,' maxcost exceeded on write to costbuf1 in fungroup '
                  stop
               end if
               write (costbuf1(indb2),31) cofg, cpay, opr, cbf, cact, cw, cpig,
     +            original_costs(opr)
            end if
         end do
      else if (cpay.eq.2) then
         do fun = 1,numfun
            if (fun_costs(fun).gt.0) then
               indb2 = indb2 + 1
               if (indb2.gt.maxcost) then
                  print *,' maxcost exceeded on write to costbuf1 in fungroup '
                  stop
               end if
               write (costbuf1(indb2),31) cofg, cpay, fun, cbf, cact, cw, cpig,
     +            fun_costs(fun)
            end if
         end do
      end if

C     Set up next group from last record read

         same = .true.
         do opr = 1, numopr
            original_costs(opr) = 0.0
         end do
         cofg = ofg
         cpay = pay
         cbf = bf
         cact = act
         cw = w
         cpig = pig
         original_costs(copr) = original_costs(copr) + cost

      end do

      if (debug) print *,' number of records written to costbuf1 = ',indb2
      numout = indb2

      return
      end

C -------------------------------------------------------------------------------
```

```
      subroutine sortcost(n)

c     Purpose:  Sorts records for level 1 cost distribution

      implicit none

      include 'liocatt.h'

      integer*4   i, j, l, n, ir
      character*22   rra

      if (debug) print *,' entering sortcost '

      l=n/2+1
      ir=n
10    continue

      if(l.gt.1)then
         l=l-1
         rra=costbuf1(l)
      else
         rra=costbuf1(ir)
         costbuf1(ir)=costbuf1(1)
         ir=ir-1
         if(ir.eq.1)then
            costbuf1(1)=rra
            if (debug) print *,' sortcost finished '
            return
         end if
      end if
      i=l
      j=l+l
20    if (j.le.ir) then
         if (j.lt.ir) then
            if (costbuf1(j)(1:14).lt.costbuf1(j+1)(1:14)) j=j+1
         end if
         if (rra(1:14).lt.costbuf1(j)(1:14)) then
            costbuf1(i)=costbuf1(j)
            i=j
            j=j+j
         else
            j=ir+1
         end if
         goto 20
      end if
      costbuf1(i)=rra
      goto 10

      end
```

```
      subroutine level1(numin,numouta,numoutb)

C     PURPOSE: Perform level one distribution of mixed mail costs to
C              direct mail codes.

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   maxgrp

      parameter   (maxgrp = 20000)

      integer*4   size1
      parameter   (size1 = numact*nummix)

      integer*2   group(4,maxgrp)
      integer*4   actptr(2,numact,numbf)
      integer*4   bfptr(2,numbf)

      real*8      original_costs(maxgrp)
      real*8      dist_mix_costs(npig,maxgrp)
      real*8      sum, cost, mixsum, chkmix

      integer*4   numin, numouta, numoutb
      integer*4   indin, inda, indb, indx
      integer*2   cofg, cpay, copr, cbf, cact, cw, cpig
      integer*2   ofg, pay, opr, bf, act, mixkey, w, ind, pig
      integer*4   i, j, k
      integer*4   ier/0/

      logical     same

      logical   debug1/.true./

      if (debug) print *,' Entering Level1.f77 '

31    format(7a2,a8)

C     Perform level one allocation
C
C       1. Collect matrix of costs for a office group, pay and cost group
C          category cell
C       2. Within each basic function cell :
C          a. For each mixed mail activity sum over direct mail costs it is
C             to be distributed to.
C          b. If sum is positive use shares to distribute mixed mail costs to
C             to direct mail costs.
C          c. If sum is zero add mixed costs to same cell for basic function 4
C          d. Output records for this bf cell.
C              1) all direct costs and all bf 4 costs to "a" file
C              2) all distributed direct cost to "b" file (bfs 1-3)

C       Set up matrices for first record

      do bf = 1, numbf          ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do
      indin = 1
      inda = 0
      indb = 0

      same = .true.
      ind = 1

C       Read first record of first cost group

      read (costbuf1(indin),31) cofg, cpay, copr, cbf, cact, cw, cpig, cost

        iginal_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
```

```
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1
      ier = 0

      do while (ier.eq.0)

C          Read rest of cost group

         do while (same)
            indin = indin + 1
            if (indin.gt.numin) then
               ier = -1
               goto 100
            end if
            read (costbuf1(indin),31) ofg, pay, opr, bf, act, w, pig, cost
            if ((ofg.eq.cofg).and.(pay.eq.cpay).and.(opr.eq.copr)) then
               ind = ind + 1
               if (debug) then
                  if (ind.gt.maxgrp) then
                     print *,' maxgrp exeeeded , ofg = ',ofg,' pay = ',pay,' opr = ',opr
                     ier = -999
                     goto 100
                  end if
               end if
               original_costs(ind) = cost
               group(1,ind) = bf
               group(2,ind) = act
               group(3,ind) = w
               group(4,ind) = pig
               if (actptr(1,act,bf).eq.0) then
                  actptr(1,act,bf) = ind
                  actptr(2,act,bf) = 1
               else
                  actptr(2,act,bf) = actptr(2,act,bf) + 1
               end if
               if (bfptr(1,bf).eq.0) then
                  bfptr(1,bf) = ind
                  bfptr(2,bf) = 1
               else
                  bfptr(2,bf) = bfptr(2,bf) + 1
               end if
            else
               same = .false.
               cbf = bf
               cact = act
               cw = w
               cpig = pig
            end if
         end do
100      if ((ier.ne.0).and.(debug))  print *,' Read exit code = ',ier
         do i = 1, ind
            do j = 1, npig
               dist_mix_costs(j,i) = 0.0
            end do
         end do
         if (debug1) then
            print *,' bfptr(1,1) = ',bfptr(1,1)
            print *,' bfptr(1,2) = ',bfptr(1,2)
            print *,' bfptr(1,3) = ',bfptr(1,3)
            print *,' bfptr(1,4) = ',bfptr(1,4)
         end if

C          Attempt to distribute mixed dollars into direct costs

         do bf = 1,3            ! do not attempt to distribute other at this level
            if (bfptr(1,bf).ne.0) then
               do i = 1,nummix  ! loop over mixed mail activities
                  if (actptr(1,mix_to_act(i),bf).gt.0) then
                     do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                        sum = 0
                        mixsum = original_costs(indx)
                        pig = group(4,indx)
                        do j = 1,count(i) ! sum over direct keys for mixed code
                           mixkey = mixmap(j,i)
                           if (actptr(1,mixkey,bf).ne.0) then
                              do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                                 sum = sum + original_costs(k)
                              end do
                           end if
```

```
                  end do
                  chkmix = 0
                  if (sum.gt.0) then ! distribute to direct codes
                     do j = 1,count(i)
                        mixkey = mixmap(j,i)
                        if (actptr(1,mixkey,bf).ne.0) then
                           do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                              dist_mix_costs(pig,k) = dist_mix_costs(pig,k) +
     +                           mixsum*(original_costs(k)/sum)
                              if (debug1) chkmix = chkmix +
     +                           mixsum*(original_costs(k)/sum)
                           end do
                        end if
                     end do
                     original_costs(indx) = 0.0
                     if (dabs(mixsum-chkmix).gt.1.0)
     &                  print *,' allocation failure, mixsum = ',mixsum,', chkmix = ',chkmix
                  end if
               end do
            end if
         end do
         do k = bfptr(1,bf), (bfptr(1,bf)+bfptr(2,bf)-1) ! Output records for this opr,bf  cell
            if (original_costs(k).gt.0.0) then
               inda = inda + 1
               write (costbuf2(inda),31) cofg, cpay, copr,
     +            group(1,k), group(2,k), group(3,k), group(4,k), original_costs(k)
            end if
            do pig = 1, npig
               if (dist_mix_costs(pig,k).gt.0.0) then
                  indb = indb + 1
                  if (indb.le.maxl1b) then
                     write (level1b(indb),31) cofg, cpay, copr, group(1,k),
     +                  group(2,k), group(3,k), pig, dist_mix_costs(pig,k)
                  else
                     print *,' maxl1b exceeded, inda = ', inda
                     stop
                  end if
               end if
            end do
         end do
      end if
   end do
   if (bfptr(1,4).gt.0) then
      do k = bfptr(1,4), ind ! Output all records for basic function "other"
         inda = inda + 1
         write (costbuf2(inda),31) cofg, cpay, copr, group(1,k),
     +      group(2,k), group(3,k), group(4,k), original_costs(k)
      end do
   end if

C     Set up next cost group using last record read

   if (ind.gt.(numbf*numact)) then
      do bf = 1, numbf    ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
   else
      do k = 1, ind
         bf = group(1,k)
         act = group(2,k)
         actptr(1,act,bf) = 0
      end do
   end if
   do bf = 1, numbf
      bfptr(1,bf) = 0
   end do

   same = .true.
   ind = 1
   cofg = ofg
   copr = opr
   cpay = pay
   cpig = pig

   original_costs(ind) = cost
   group(1,ind) = cbf
   group(2,ind) = cact
   group(3,ind) = cw
```

```
        actptr(1,cact,cbf) = ind
        actptr(2,cact,cbf) = 1
        bfptr(1,cbf) = ind
        bfptr(2,cbf) = 1

      end do

       umouta = inda
      numoutb = indb

      if (debug) print *,' number of records written to costbuf2 = ',numouta
      if (debug) print *,' number of records written to level1b  = ',numoutb
      return
      end

C -----------------------------------------------------------------------
```

```
      subroutine level2(numin,numouta,numoutb)

C     PURPOSE: Perform level two distribution of mixed mail costs to
C              direct mail codes.

      IMPLICIT NONE

      include 'liocatt.h'

      integer*4   maxgrp

      parameter   (maxgrp = 20000)

      integer*4   size1
      parameter   (size1 = numact*nummix)

      integer*2   group(4,maxgrp)
      integer*4   actptr(2,numact,numbf)
      integer*4   bfptr(2,numbf)

      real*8      original_costs(maxgrp)
      real*8      dist_mix_costs(npig,maxgrp)
      real*8      sum, cost, mixsum, chkmix
      real*8      mixpig(npig)
      real*8      tdirect/0/  , tmixed/0/ , tmixeddist/0/ , tratio/0/

      integer*4   numin, numouta, numoutb
      integer*4   indin, inda, indb, indx
      integer*2   cofg, cpay, copr, cbf, cact, cw, cpig
      integer*2   ofg, pay, opr, bf, act, mixkey, w, ind, pig
      integer*2   bf4/4/
      integer*4   i, j, k
      integer*4   ier/0/

      logical     same , dist
      logical   debug1/.true./

      if (debug) print *,' Entering Level2.f77 '

C     open input and output file

31    format(7a2,a8)
45    format(7a2,a8)

C       Perform level two mixed cost allocation
C
C       1. Collect matrix of costs for a office group,pay category, and
C          operation/route code cell
C       2. Over all records in cell
C          a. For each mixed mail activity sum over basic functions to get
C             mixed mail costs.
C          c. For each mixed mail activity sum over direct mail costs it is
C             to be distributed to (over all basic functions).
C          b. If sum is positive use shares to distribute mixed mail costs to
C             to direct mail costs (all distributed mixed costs get basic
C             function 4.
C          d. Output records for this opr cell.
C              1) all direct costs costs to "a" file
C              2) all distributed direct cost to "b" file (bf 4)

C       Set up matrices for first record

      do bf = 1, numbf          ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do
      indin = 1
      inda = 0
      indb = 0

        me = .true.
       id = 1

        Read first record of first cost group

      read (costbuf2(indin),31) cofg, cpay, copr, cbf, cact, cw, cpig, cost
```

```
      original_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
       ctptr(1,cact,cbf) = ind
       ctptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1
      ier = 0

      do while (ier.eq.0)

C           Read rest of cost group

         do while (same)
            indin = indin + 1
            if (indin.gt.numin) then
               ier = -1
               goto 100
            end if
            read (costbuf2(indin),31) ofg, pay, opr, bf, act, w, pig, cost
            if ((ofg.eq.cofg).and.(pay.eq.cpay).and.(opr.eq.copr)) then
               ind = ind + 1
               if (debug) then
                  if (ind.gt.maxgrp) then
                     print *,' maxgrp exeeeded , ofg = ',ofg,' pay = ',pay,' opr = ',opr
                     ier = -999
                     goto 100
                  end if
               end if
               original_costs(ind) = cost
               group(1,ind) = bf
               group(2,ind) = act
               group(3,ind) = w
               group(4,ind) = pig
               if (actptr(1,act,bf).eq.0) then
                  actptr(1,act,bf) = ind
                  actptr(2,act,bf) = 1
               else
                  actptr(2,act,bf) = actptr(2,act,bf) + 1
               end if
               if (bfptr(1,bf).eq.0) then
                  bfptr(1,bf) = ind
                  bfptr(2,bf) = 1
               else
                  bfptr(2,bf) = bfptr(2,bf) + 1
               end if
            else
               same = .false.
               cbf = bf
               cact = act
               cw = w
               cpig = pig
            end if
         end do
100      if ((ier.ne.0).and.(debug)) print *,' Read exit code = ',ier
         do i = 1, ind
            do j = 1, npig
               dist_mix_costs(j,i) = 0.0
            end do
         end do
         if (debug1) then
            print *,' bfptr(1,1) = ',bfptr(1,1)
            print *,' bfptr(1,2) = ',bfptr(1,2)
            print *,' bfptr(1,3) = ',bfptr(1,3)
            print *,' bfptr(1,4) = ',bfptr(1,4)
         end if

C           Attempt to distribute mixed dollars into direct costs

         do i = 1,nummix          ! loop over mixed mail activities
            do pig = 1, npig
               mixpig(pig) = 0.0
            end do
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                     pig = group(4,indx)
```

```
                  mixpig(pig) = mixpig(pig) + original_costs(indx)
               end do
            end if
         end do
         dist = .false.
         do pig = 1, npig
            if (mixpig(pig).gt.0.0) then
               sum = 0.0
               do bf = 1, numbf
                  do j = 1,count(i) ! sum over direct keys for mixed code
                     mixkey = mixmap(j,i)
                     if (actptr(1,mixkey,bf).ne.0) then
                        do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                           sum = sum + original_costs(k)
                        end do
                     end if
                  end do
               end do
               chkmix = 0
               if (sum.gt.0) then ! distribute to direct codes
                  do bf = 1, numbf
                     do j = 1,count(i)
                        mixkey = mixmap(j,i)
                        if (actptr(1,mixkey,bf).ne.0) then
                           do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                              dist_mix_costs(pig,k) = dist_mix_costs(pig,k) +
     +                           mixpig(pig)*(original_costs(k)/sum)
                              if (debug1) chkmix = chkmix +
     +                           mixpig(pig)*(original_costs(k)/sum)
                           end do
                        end if
                     end do
                  end do
                  dist = .true.
                  if (dabs(mixpig(pig)-chkmix).gt.1.0)
     &                print *,' level2 allocation failure, mixsum = ',mixsum,', chkmix = ',chkmix
               end if
            end if
         end do
         if (dist) then
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                     original_costs(indx) = 0.0
                  end do
               end if
            end do
         end if
      end do
      do k = 1, ind
         if (original_costs(k).gt.0.0) then
            inda = inda + 1
            write (costbuf1(inda),45) cofg, cpay, copr, group(1,k),
     +         group(2,k), group(3,k), group(4,k), original_costs(k)
         end if
         do pig = 1, npig
            if (dist_mix_costs(pig,k).gt.0.0) then
               indb = indb + 1
               if (indb.le.maxl2b) then
                  write (level2b(indb),45) cofg, cpay, copr, bf4, group(2,k),
     +                group(3,k), pig, dist_mix_costs(pig,k)
               else
                  print *,' maxl2b exceeded, inda = ',inda
                  stop ' fatal error '
               end if
            end if
         end do
      end do

C     Set up next cost group using last record read

      if (ind.gt.(numbf*numact)) then
         do bf = 1, numbf    ! initialize actptr array
            do act = 1, numact
               actptr(1,act,bf) = 0
            end do
         end do
      else
         do k = 1, ind
            bf = group(1,k)
```

```
            act = group(2,k)
            actptr(1,act,bf) = 0
         end do
      end if
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do

      same = .true.
      ind = 1
      cofg = ofg
      copr = opr
      cpay = pay

      original_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1

   end do

   numouta = inda
   numoutb = indb

   if (debug) print *,' number of records written to costbuf1 = ',numouta
   if (debug) print *,' number of records written to level2b  = ',numoutb
   return
   end

C ------------------------------------------------------------------------------
```

```
      subroutine sortlev2a(n)

C     Purpose:  To sort records for level 3 cost distribution

      implicit none

      include 'liocatt.h'

      integer*4   i, j, l, n, ir
      character*22   rra

      if (debug) print *,' entering sortlev2a.f77 '

      l=n/2+1
      ir=n
10    continue

      if(l.gt.1)then
         l=l-1
         rra=costbuf1(l)
      else
         rra=costbuf1(ir)
         costbuf1(ir)=costbuf1(1)
         ir=ir-1
         if(ir.eq.1)then
            costbuf1(1)=rra
            if (debug) print *,' exiting sortlev2a '
            return
         end if
      end if
      i=l
      j=l+l
20    if (j.le.ir) then
         if (j.lt.ir) then
            if (costbuf1(j)(3:12).lt.costbuf1(j+1)(3:12)) j=j+1
         end if
         if (rra(3:12).lt.costbuf1(j)(3:12)) then
            costbuf1(i)=costbuf1(j)
            i=j
            j=j+j
         else
            j=ir+1
         end if
         goto 20
      end if
      costbuf1(i)=rra
      goto 10
      end
```

```
      subroutine level3(numin,numouta,numoutb)

C     PURPOSE: Perform level three distribution of mixed mail costs to direct mail codes.

      IMPLICIT NONE

       include 'liocatt.h'

      integer*4   maxgrp

      parameter   (maxgrp = 200000)

      integer*4   sizel
      parameter   (sizel = numact*nummix)

      integer*2   group(4,maxgrp)
      integer*4   actptr(2,numact,numbf)
      integer*4   bfptr(2,numbf)

      real*8      original_costs(maxgrp)
      real*8      dist_mix_costs(npig,maxgrp)
      real*8      sum, cost, mixsum, chkmix
      real*8      mixed(npig)

      integer*4   numin, numouta, numoutb
      integer*4   indin, inda, indb, indx
      integer*2   cpay, copr, cbf, cact, cw, cpig
      integer*2   pay, opr, bf, act, mixkey, w, ind, pig
      integer*2   bf4/4/
      integer*4   i, j, k
      integer*4   ier/0/

      logical     same , dist
      logical   debug1/.true./

      if (debug) print *,' entering level3.f77 '

C       open input and output file

31    format(2x,6a2,a8)
45    ormat(6a2,a8)

C       Perform level three mixed cost allocation
C
C       1. Collect matrix of costs for a pay category,operation/route code cell
C       2. Over all records in cell
C          a. For each mixed mail activity sum over basic functions to get
C             mixed mail costs.
C          c. For each mixed mail activity sum over direct mail costs it is
C             to be distributed to (over all basic functions).
C          b. If sum is positive use shares to distribute mixed mail costs to
C             to direct mail costs (all distributed mixed costs get basic
C             function 4.
C          d. Output records for this opr cell.
C              1) all direct costs costs to "a" file
C              2) all distributed direct cost to "b" file (bf 4)

C       Set up matrices for first record

      do bf = 1, numbf          ! initialize actptr array
         do act = 1, numact
            actptr(1,act,bf) = 0
         end do
      end do
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do
      indin = 1
      inda = 0
      indb = 0

      same = .true.
      ind = 1

C       Read first record of first cost group

      read (costbuf1(indin),31) cpay, copr, cbf, cact, cw, cpig, cost

      original_costs(ind) = cost
      group(1,ind) = cbf
```

```
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1
      ier = 0

      do while (ier.eq.0)

C          Read rest of cost group

         do while (same)
            indin = indin + 1
            if (indin.gt.numin) then
               ier = -1
               goto 100
            end if
            read (costbuf1(indin),31) pay, opr, bf, act, w, pig, cost
            if ((pay.eq.cpay).and.(opr.eq.copr)) then
               if ((bf.eq.group(1,ind)).and.
     +             (act.eq.group(2,ind)).and.
     +             (w.eq.group(3,ind)).and.
     +             (pig.eq.group(4,ind))) then
                  original_costs(ind) = original_costs(ind) + cost
               else
                  ind = ind + 1
                  if (debug) then
                     if (ind.gt.maxgrp)
     +                  print *,' maxgroup exceeded, pay = ',pay,', opr = ',opr
                  end if
                  original_costs(ind) = cost
                  group(1,ind) = bf
                  group(2,ind) = act
                  group(3,ind) = w
                  group(4,ind) = pig
                  if (actptr(1,act,bf).eq.0) then
                     actptr(1,act,bf) = ind
                     actptr(2,act,bf) = 1
                  else
                     actptr(2,act,bf) = actptr(2,act,bf) + 1
                  end if
                  if (bfptr(1,bf).eq.0) then
                     bfptr(1,bf) = ind
                     bfptr(2,bf) = 1
                  else
                     bfptr(2,bf) = bfptr(2,bf) + 1
                  end if
               end if
            else
               same = .false.
               cbf = bf
               cact = act
               cw = w
               cpig = pig
            end if
         end do
100      if ((ier.ne.0).and.debug) print *,' Read exit code = ',ier
         do i = 1, ind
            do pig = 1, npig
               dist_mix_costs(pig,i) = 0.0
            end do
         end do
         if (debug1) then
            print *,' bfptr(1,1) = ',bfptr(1,1)
            print *,' bfptr(1,2) = ',bfptr(1,2)
            print *,' bfptr(1,3) = ',bfptr(1,3)
            print *,' bfptr(1,4) = ',bfptr(1,4)
         end if

C          Attempt to distribute mixed dollars into direct costs

         do i = 1,nummix        ! loop over mixed mail activities
            do pig = 1, npig
               mixed(pig) = 0.0
            end do
            do bf = 1, numbf
               if (actptr(1,mix_to_act(i),bf).gt.0) then
                  do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
```

```
                pig = group(4,indx)
                mixed(pig) = mixed(pig) + original_costs(indx)
             end do
          end if
       end do
       dist = .false.
       do pig = 1, npig
          if (mixed(pig).gt.0.0) then
             sum = 0.0
             do bf = 1, numbf
                do j = 1,count(i) ! sum over direct keys for mixed code
                   mixkey = mixmap(j,i)
                   if (actptr(1,mixkey,bf).ne.0) then
                      do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                         sum = sum + original_costs(k)
                      end do
                   end if
                end do
             end do
             chkmix = 0
             if (sum.gt.0) then ! distribute to direct codes
                do bf = 1, numbf
                   do j = 1,count(i)
                      mixkey = mixmap(j,i)
                      if (actptr(1,mixkey,bf).ne.0) then
                         do k = actptr(1,mixkey,bf),(actptr(1,mixkey,bf)+actptr(2,mixkey,bf)-1)
                            dist_mix_costs(pig,k) = dist_mix_costs(pig,k) +
+                              mixed(pig)*(original_costs(k)/sum)
                            if (debug1) chkmix = chkmix +
+                              mixed(pig)*(original_costs(k)/sum)
                         end do
                      end if
                   end do
                   if (actptr(1,mix_to_act(i),bf).ne.0)
+                      original_costs(actptr(1,mix_to_act(i),bf)) = 0.0
                end do
                dist = .true.
             end if
             if (dabs(mixed(pig)-chkmix).gt.1.0)
&               print *,' level 3 allocation failure, mixsum = ',mixsum,', chkmix = ',chkmix
          end if
       end do
       if (dist) then
          do bf = 1, numbf
             if (actptr(1,mix_to_act(i),bf).gt.0) then
                do indx = actptr(1,mix_to_act(i),bf),(actptr(1,mix_to_act(i),bf)+actptr(2,mix_to_act(i),bf)-1)
                   original_costs(indx) = 0.0
                end do
             end if
          end do
       end if
    end do
    do k = 1, ind
       if (original_costs(k).gt.0.0) then
          inda = inda + 1
          write (costbuf2(inda),45) cpay, copr, group(1,k),
+            group(2,k), group(3,k), group(4,k), original_costs(k)
       end if
       do pig = 1, npig
          if (dist_mix_costs(pig,k).gt.0.0) then
             indb = indb + 1
             if (indb.le.maxl3b) then
                write (level3b(indb),45) cpay, copr, bf4, group(2,k),
+                  group(3,k), pig, dist_mix_costs(pig,k)
             else
                print *,' maxl3b exceeded inda = ',inda
                stop
             end if
          end if
       end do
    end do

C      Set up next cost group using last record read

    if (ind.gt.(numbf*numact)) then
       do bf = 1, numbf    ! initialize actptr array
          do act = 1, numact
             actptr(1,act,bf) = 0
          end do
       end do
```

```
      else
         do k = 1, ind
            bf = group(1,k)
            act = group(2,k)
            actptr(1,act,bf) = 0
         end do
      end if
      do bf = 1, numbf
         bfptr(1,bf) = 0
      end do

      same = .true.
      ind = 1
      cpay = pay
      copr = opr

      original_costs(ind) = cost
      group(1,ind) = cbf
      group(2,ind) = cact
      group(3,ind) = cw
      group(4,ind) = cpig
      actptr(1,cact,cbf) = ind
      actptr(2,cact,cbf) = 1
      bfptr(1,cbf) = ind
      bfptr(2,cbf) = 1

   end do

   numouta = inda
   numoutb = indb

   if (debug) print *,' number of records written to level3a = ',numouta
   if (debug) print *,' number of records written to level3b = ',numoutb
   return
   end

C ------------------------------------------------------------------------------
```

```
      subroutine report(nlev1b,nlev2b,nlev3a,nlev3b)

C     PURPOSE: Produce report on results of Liocatt by activity for city carriers by
C              activity code and operation/route code

      IMPLICIT NONE

      include 'liocatt.h'

      real*8       data(numw,numact)
      real*8       indata

      integer*4    nlev1b, nlev2b, nlev3a, nlev3b, irun
      integer*2    ofg, opr, act, pay, bf, w, pig
      integer*4    i, j
      integer*4    ier/0/

      character*3 buffer

      logical flag/.false./

      if (debug) then
         print *,' in subroutine report '
         print *, ' nlev1b = ',nlev1b,' nlev2b = ',nlev2b
         print *, ' nlev3a = ',nlev3a,' nlev3b = ',nlev3b
      end if
      do i = 1, numact
         do j = 1, numw
            data(j,i) = 0.0
         end do
      end do

C       Open input files

25    format(7a2,a8)
35    format(6a2,a8)

      open(20,file='level1b')
      open(21,file='level2b')
      open(22,file='level3a')
      open(23,file='level3b')
45    format(i2.2,i1,i2.2,i1,i3.3,i3.3,i2,f13.1)
46    format(i1,i2.2,i1,i3.3,i3.3,i2,f13.1)

C       Assemble data for report

      do i = 1, nlev1b
         read (level1b(i),25) ofg,pay,opr,bf,act,w,pig,indata
         write (20,45) ofg,pay,opr,bf,act,w,pig,indata
      end do
      do i = 1, nlev2b
         read (level2b(i),25) ofg,pay,opr,bf,act,w,pig,indata
         write (21,45) ofg,pay,opr,bf,act,w,pig,indata
      end do
      do i = 1, nlev3a
         read (costbuf2(i),35) pay,opr,bf,act,w,pig,indata
         write (22,46) pay,opr,bf,act,w,pig,indata
      end do
      do i = 1, nlev3b
         read (level3b(i),35) pay,opr,bf,act,w,pig,indata
         write (23,46) pay,opr,bf,act,w,pig,indata
      end do

      return
      end

C -------------------------------------------------------------------------------
```

```
      PROGRAM rpt_wgt22cra

c     Purpose:  To summarize city carrier in-office costs by weight increment, shape,
C               and CRA subclass

      IMPLICIT NONE

      integer*4   numfun, numact, nopr, nshp, numfun2, ncl

      parameter   (numfun = 22) ! number of weight increments
      parameter   (numfun2 = 21) !  number of weight increments (exclude no weight)
      parameter   (numact = 501) ! number of activity codes
      parameter   (nopr = 12)   !  number of operations
      parameter   (ncl = 243)   !  number of activity codes
      parameter   (nshp = 4)    !  number of shapes

      integer*4   is, shape, ishp, shp(numact), icl
      integer*4   pay, opr, bf, act, w
      integer*4   unit, i, ier
      integer*4   class(numact)

      real*8      carrier(ncl,nshp,numfun)
      real*8      indata, cardist(ncl,nshp,numfun), sum
      real*8      actwgt(numfun)

      character*4 acodes(numact)
      character*9 classes(ncl)
      character*4 shapetype(nshp) /'1Ltr ','2Flt ','3Pcl ','4None'/


      ier = 0

c     Map of activity codes and codes to corresponding subclass
      open(16,file='activity00.ecr.all')
17    format(a4,i6)

      do i = 1,numact
         read(16,17) acodes(i), class(i)
         is = shape(acodes(i))
         shp(i) = is
      end do
      close(16)

c     Map of subclasses
      open(16,file='classes_ecr.old')
18    format(a9)
      do i = 1, ncl
         read(16,18) classes(i)
      end do
      close(16)

c     Initialize matrices
      do icl = 1, ncl
         do ishp = 1, nshp
            do i = 1, numfun
               carrier(icl,ishp,i) = 0.0
               cardist(icl,ishp,i) = 0.0
            end do
         end do
      end do

c     Open files of LIOCATT results
      open(20,file='level1b')
      open(21,file='level2b')
25    format(2x,i1,i2.2,i1,2i3.3,2x,f13.1)
      open(30,file='level3a')
      open(31,file='level3b')
35    format(i1,i2.2,i1,2i3.3,2x,f13.1)

      do unit = 20,21
         do while (ier.eq.0)
            read (unit,25,iostat=ier,end=100) pay,opr,bf,act,w,indata
            icl = class(act)     !  Assign subclass
            ishp = shp(act)      !  Assign shape
            if (icl.eq.2) icl = 1 !  Combine 1st single piece
            if ((icl.ge.3).and.(icl.le.4)) icl = 5 !  Combine 1st presort
            if (icl.eq.7) icl = 6 !  Combine 1st single piece cards
            if ((icl.eq.8).or.(icl.eq.9)) icl = 10 !  Combine 1st presort cards
            if ((icl.ge.19).and.(icl.le.21)) icl = 18 !  Combine ECR
            if (icl.eq.22) icl = 23 !  Combine Std A Reg
```

```
         if ((icl.ge.25).and.(icl.le.27)) icl = 24 !  Combine NP ECR
         if (icl.eq.28) icl = 29 !  Combine Std A NP
         if (icl.eq.31) icl = 30 !  Combine 4th parcel post
         if ((icl.ge.38).and.(icl.le.70)) icl = 37 !  Combine intl
         if (pay.eq.3) then  !  City carriers
            if (icl.gt.0) then
               if (ishp.gt.0) then
                  carrier(icl,ishp,w) = carrier(icl,ishp,w) + indata/1000.
               else
                  print*, 'Invalid shape ', acodes(act), shp(act)
               end if
            else
               print*, 'Invalid class assignment ', acodes(act), class(act)
            end if
         end if
      end do
100   print *,' Read exit of unit ',unit,' = ',ier
      ier = 0
   end do

   ier = 0

   do unit = 30,31
      do while (ier.eq.0)
         read (unit,35,iostat=ier,end=101) pay,opr,bf,act,w,indata
         icl = class(act)     !  Assign subclass
         ishp = shp(act)      !  Assign shape
         if (icl.eq.2) icl = 1 !  Combine 1st single piece
         if ((icl.ge.3).and.(icl.le.4)) icl = 5 !  Combine 1st presort
         if (icl.eq.7) icl = 6 !  Combine 1st single piece cards
         if ((icl.eq.8).or.(icl.eq.9)) icl = 10 !  Combine 1st presort cards
         if ((icl.ge.19).and.(icl.le.21)) icl = 18 !  Combine ECR
         if (icl.eq.22) icl = 23 !  Combine Std A Reg
         if ((icl.ge.25).and.(icl.le.27)) icl = 24 !  Combine NP ECR
         if (icl.eq.28) icl = 29 !  Combine Std A NP
         if (icl.eq.31) icl = 30 !  Combine 4th parcel post
         if ((icl.ge.38).and.(icl.le.70)) icl = 37 !  Combine intl
         if (pay.eq.3) then  !  City carriers
            if (icl.gt.0) then
               if (ishp.gt.0) then
                  carrier(icl,ishp,w) = carrier(icl,ishp,w) + indata/1000.
               else
                  print*, 'Invalid shape ', acodes(act), shp(act)
               end if
            else
               print*, 'Invalid class assignment ', acodes(act), class(act)
            end if
         end if
      end do
101   print *,' Read exit of unit ',unit,' = ',ier
      ier = 0
   end do

c  Redistribute no weight city carrier costs
   do icl = 1, 37            !  Skip Spec Serv subclasses
      do ishp = 1, nshp
         sum = 0.0
         if (carrier(icl,ishp,numfun).gt.0.0) then
            do i = 1, numfun2 !  Distribute over all weight increments
               sum = sum + carrier(icl,ishp,i)
            end do
            if (sum.gt.0.0) then
               do i = 1, numfun2
                  cardist(icl,ishp,i) = cardist(icl,ishp,i) +
     &                (carrier(icl,ishp,numfun)*(carrier(icl,ishp,i)/sum))
               end do
               carrier(icl,ishp,numfun) = 0.0
            else
               sum = 0.0
               do i = 1, numfun2
                  actwgt(i) = 0.0
               end do
               if (carrier(icl,ishp,numfun).gt.0.0) then
                  do i = 1, numfun2 !  Redistribute over all weight increments
                     actwgt(i) = actwgt(i) + carrier(icl,ishp,i)
                     sum = sum + carrier(icl,ishp,i)
                  end do
                  if (sum.gt.0.0) then
                     do i = 1, numfun2
                        cardist(icl,ishp,i) = cardist(icl,ishp,i) +
```

```
     &                        (carrier(icl,ishp,numfun)*(actwgt(i)/sum))
                           end do
                           carrier(icl,ishp,numfun) = 0.0
                        else
                           print*, 'Unable to distribute no weight costs for class ', classes(icl)
                        end if
                     end if
                  end if
               end if
            end do
         end do

         print*, 'Costs redistributed '

c        Add in redistributed no weight costs
         do icl = 1, ncl
            do ishp = 1, nshp
               do i = 1, numfun
                  carrier(icl,ishp,i) = carrier(icl,ishp,i) + cardist(icl,ishp,i)
               end do
            end do
         end do
         print*, 'Redistributed costs added in '

         classes(5)  = 'PreL'
         classes(10) = 'PreC'
         classes(18) = 'BRCRT'
         classes(23) = 'BRO'
         classes(24) = 'NPCRT'
         classes(29) = 'NPO'

c        Write out results
         open(45,file='car_wgt22_00cra2.csv',recl=500)
41       format(i3,',',a9,',',i2,',',a4,',',22(f12.0,','))
         do icl = 1, ncl
            do ishp = 1, nshp
               if ((icl.eq.1).or.(icl.eq.5).or.((icl.ge.15).and.(icl.le.18)).or.
     &             (icl.eq.23).or.(icl.eq.24).or.(icl.eq.29)) then
                  write (45,41) icl, classes(icl), ishp, shapetype(ishp),
     &                 (carrier(icl,ishp,i), i = 1, numfun)
               end if
            end do
         end do

         end
c--------------------------------------------------------------------------------
c        Assign shape

         function shape(act)

         integer*4    shape
         character*4  act

         if (act(1:1).eq.'1') then
            shape = 1  !  Letters
         else if (act(1:1).eq.'2') then
            shape = 2  !  Flats
         else if (act(1:1).eq.'3') then
            shape = 3  !  IPPs
         else if (act(1:1).eq.'4') then
            shape = 3  !  Parcels
         else
            shape = 4  !  Other (special services)
         end if

         return
         end
```