Learning outcomes:

1. Define accumulating total and sub-totals for groups in DATA step
2. BY-group concept
3. First. and Last.variable processing

SAS components learnt:

1. RETAIN statement
2. SUM statement
3. FIRST.*variable* and LAST.*variable*

8.1 Accumulating total in DATA step

- Create an accumulating variable
  - Electricity data revisit
    - Eight years electricity data with electricity consumption, cost, average temperature and number of heating days are stored in the SAS data set, "d:\sas_datasets\electricity.sas7bdat.".

Ex. 8.1: Construct a SAS program so that only the year 1 electricity data is extracted. Save the data in the data set, "Elect.Year1_elect".

- We create an accumulating variable called TOTAL_COST for the total cost of Year 1. Here are the SAS program and output:

```
data Year1_elect0;
   set Elect.Year1_elect;
   total_cost=total_cost+cost;
run;

proc print data=Year1_elect0;
var cost total_cost;
run;
```

```
                The SAS System        16:56 Thursday, August 9, 2012   1

                                 total_
            Obs      Cost         cost

             1     295.33          .
             2     230.08          .
             3     213.43          .
             4     338.16          .
             5     299.76          .
             6     214.44          .
```

- TOTAL_COST is always missing for all iteration of DATA step.
- Note that TOTAL_COST is defined as missing in the PDV during the initialization. Therefore it should be initialized as zero in the initialization stage.

- RETAIN statement
  - Syntax

```
RETAIN var-1 <initial value for var-1> … var-n <initial value for var-n>;
```

- *Var-1* refers to name of first variable and *var-n* refers to name of *n*-th variable.
- The statement retains the value of the variable in the PDV across the iterations of the DATA step
- When no initial value is given, the retained variable is initialized to missing before the 1st iteration of the DATA step
- It is a COMPILE-TIME only statement.

- Electricity data revisit
  - RETAIN statement is added in the SAS program and the output:

```
data Year1_elect1;
  set Elect.Year1_elect;
  retain total_cost 0;
  total_cost=total_cost+cost;
run;

proc print data=Year1_elect1;
var cost total_cost;
run;
```

```
                    The SAS System        16:56 Thursday, August 9, 2012  15

                                      total_
Obs      Cost          cost

 1      295.33        295.33
 2      230.08        525.41
 3      213.43        738.84
 4      338.16       1077.00
 5      299.76       1376.76
 6      214.44       1591.20
```

(a) Note that the initial value of TOTAL_COST is set to zero.
(b) The accumulated total cost of year 1 is $1591.20.

- Program Data Vector processing

(a) Compilation stage:
Retain flag is added to the variable "TOTAL_COST" in the PDV.

Time_period	Electricity_ Consumption	…	Cost	Total_Cost
				Retain

(b) Execution stage
Partial PDV processing:

	PDV				
Iteration	Time_period	Electricity_ Consumption	…	Cost	Total_Cost
					Retain
1 (DATA statement: initialization of PDV)		.		.	0
1 (SET statement)	Year 1: Jan/Feb	3637	…	295.33	0
1 (TOTAL_COST assignment statement)	Year 1: Jan/Feb	3637	…	295.33	295.33
1 (RUN statement)	Year 1: Jan/Feb	3637	…	295.33	295.33 → Output to Year1_ Elect1;
2 (Not EOF, re-initialization of PDV)		.	…	.	295.33*
2 (SET statement)	Year 1: March/Apr	2888	…	230.38	295.33
2 (TOTAL_COST assignment statement)	Year 1: March/Apr	2888	…	230.38	295.33+230.38 = 525.41
2 (RUN statement)	Year 1: March/Apr	2888	…	230.38	525.41 → Output to Year1_ Elect1;
3 (Not EOF, re-initialization of PDV)		.		.	525.41*

*value is retained during re-initialisation due to the retain flag.

Ex. 8.2: Rewrite the statement "TOTAL_COST=TOTAL_COST+COST;" of the above SAS program using SUM function.

Ans:

- SUM statement
  - SUM function and its general form
    - Syntax

```
SUM(Argument-1,.., Argument-n);
```

SUM function ignores any missing values.

- General form of SUM statement

```
variable + expression;
```

(a) create a variable on the left side of the plus sign if it does not exist
(b) initialize the variable to zero before the 1st iteration of the DATA step
(c) automatically RETAIN variable
(d) adds the value of expression to the variable at execution
(e) ignore MISSING value
(f) be more efficient than a RETAIN statement along with a SUM function

- Electricity data revisit

```
/* sum statement */
data Year1_elect2;
  set Elect.Year1_elect;
  total_cost+cost;
run;

proc print data=Year1_elect2;
var cost total_cost;
run;
```

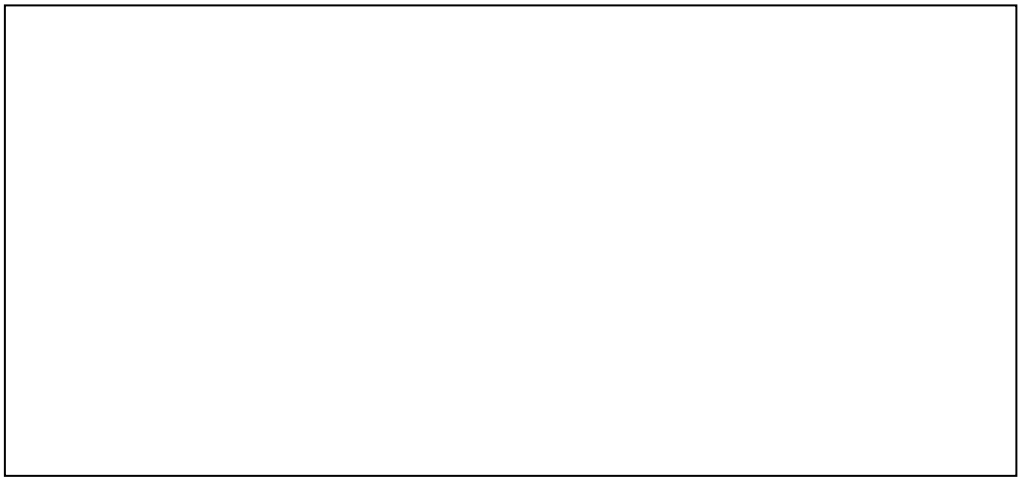
This program creates the same output as the one using RETAIN statement.

Ex. 8.3: Using the electricity data set ("D:\SAS_DATASETS\ELECTRICITY.SAS7BDAT"), calculate the average of the average temperature in Year 4:

(a) Write a SAS program to extract all observations of Year 4 and store the data in D:\SAS_DATASETS\Year4_elect.SAS7BDAT.

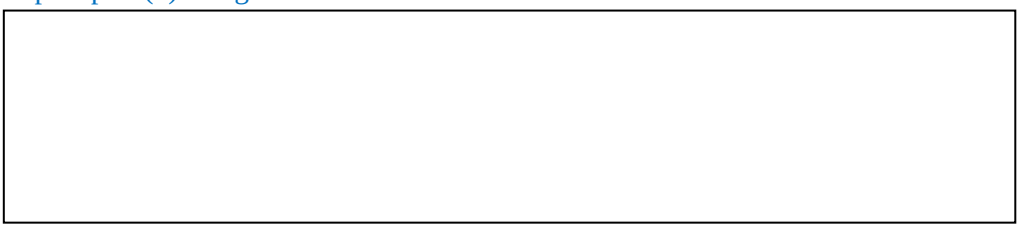
```

```

(b) Write a SAS program to calculate the average of the average temperature in Year 4 using RETAIN statement. Use PRINT procedure to get SAS output.



*Addition of a missing value to a variable gives missing value.

(c) Repeat part (b) using SUM statement.



(d) From (c), what is the average of the average temperature of Year 4?

8.2 Accumulating sub-totals for groups in DATA step

- BY-group
  - The BY statement has been introduced in Chapter 6, which provides common key(s) for merging.
  - Syntax

```
BY <DESCENDING> by-variable(s);
```

    - When BY statement used in DATA step, SORT procedure should be submitted before the DATA step is required. The sorting key is the by-variables in the DATA step. (See Chapter 6 for details).
    - Enable SAS to process data in *by-variable(s)* groups and each group has the same value of BY-variable.
  - The SORT procedure:
    - Syntax

```
PROC SORT DATA=input-SAS-data-set <OUT=sorted-SAS-data-set NODUPKEY EQUALS>;
  BY <DESCENDING> By-variable-1 … By-variable-n;
RUN;
```

    - Rearrange the observations in a data set according to the BY variables
    - Sort on single or multiple variables
    - Create a SAS data set specified in OUT option which is a sorted copy of the input SAS data set
    - Replace the input data by sorted data, by default.
  - First. and Last. Processing
    - Syntax

```
FIRST.BY-variable
LAST.BY-variable
```

(a) When a BY statement is used in DATA step, SAS creates the above two TEMPORARY variable for each variable listed in the BY statement
(b) The FIRST.BY-variable is assigned with value 1 for the FIRST observations of each BY group and 0, otherwise.
(c) Similarly, the LAST.BY-variable is assigned with value 1 for the LAST observations of each BY group and 0, otherwise.

- Single BY-variable
  - The process of summarising data
    - To obtain the subtotal, it is required to set the accumulating variable to zero for the FIRST observation for each group. An IF-THEN statement is required.
    - Use SUM statement to make increment of the accumulating variable. Note that RETAIN action is automatically done when SUM statement is used.
    - Output the value of the accumulating variable for the LAST observation of each group.
  - Electricity revisit: Yearly total electricity cost
    - The BY-variable is Period and the FIRST.*BY-variable* and LAST.*BY-variable* are assigned with 0 or 1 according to their order in within the BY-group and their values in PDV are:

Time_Period	Period	FIRST.Period	LAST.Period	Cost
Year 1: Jan/Feb	Year 1	**1**	0	295.33
Year 1: March/Apr	Year 1	0	0	230.08
Year 1: May/June	Year 1	0	0	213.43
Year 1: July/Aug	Year 1	0	0	338.16
Year 1: Sept/Oct	Year 1	0	0	299.76
Year 1: Nov/Dec	Year 1	0	**1**	214.44
Year 2: Jan/Feb	Year 2	1	0	384.13
Year 2: …	Year 2	…	…	...
Year 2: Nov/Dec	Year 2	0	1	276.13
Year 3: Jan/Feb	Year 3	1	0	321.94
Year 3: …	Year 3	…	…	...
Year 3: Nov/Dec	Year 3	0	1	183.84
…				…
Year 8: Jan/Feb	Year 8	1	0	309.4
Year 8: …	Year 8	…	…	...
Year 8: Nov/Dec	Year 8	0	1	229.05

- There are 3 steps for calculating the yearly total of the electricity cost.
- The accumulating variable used in the program is called TOTAL_COST.

```
*p8_electricity.sas;
/*Step 1: Get Year period for the data */
data Elect.electricityCYr;
length Period $8;
  set Elect.electricity;
  Period=Substr(Time_period,1,6);
run;
/*Step 2: Sort the data by period */
proc sort data=elect.electricityCyr;
by period;
run;
/*Step 3: Accumulating the total cost by years */
data electricitycyr;
   set elect.electricitycyr;
   by period;
   if First.period then total_cost=0;
   total_cost+cost;
run;
proc print data=electricitycyr;
var time_period period cost total_cost;
run;
```

- The output:

Obs	Time_Period	Period	Cost	total_ cost
1	Year 1: Jan/Feb	Year 1	295.33	295.33
2	Year 1: March/Apr	Year 1	230.08	525.41
3	Year 1: May/June	Year 1	213.43	738.84
4	Year 1: July/Aug	Year 1	338.16	1077.00
5	Year 1: Sept/Oct	Year 1	299.76	1376.76
6	Year 1: Nov/Dec	Year 1	214.44	**1591.20**
7	Year 2: Jan/Feb	Year 2	384.13	384.13
8	Year 2: March/Apr	Year 2	295.82	679.95
9	Year 2: May/June	Year 2	255.85	935.80
10	Year 2: July/Aug	Year 2	219.72	1155.52
11	Year 2: Sept/Oct	Year 2	256.59	1412.11
12	Year 2: Nov/Dec	Year 2	276.13	**1688.24**
13	Year 3: Jan/Feb	Year 3	321.94	321.94
14	Year 3: March/Apr	Year 3	221.11	543.05
15	Year 3: May/June	Year 3	205.16	748.21
16	Year 3: July/Aug	Year 3	251.07	999.28
17	Year 3: Sept/Oct	Year 3	279.80	1279.08
18	Year 3: Nov/Dec	Year 3	183.84	**1462.92**
19	Year 4: Jan/Feb	Year 4	244.93	244.93
20	Year 4: March/Apr	Year 4	218.59	463.52
21	Year 4: May/June	Year 4	213.09	676.61
22	Year 4: July/Aug	Year 4	333.49	1010.10
23	Year 4: Sept/Oct	Year 4	370.35	1380.45
24	Year 4: Nov/Dec	Year 4	222.79	**1603.24**
25	Year 5: Jan/Feb	Year 5	316.18	316.18
26	Year 5: March/Apr	Year 5	77.39	393.57
27	Year 5: May/June	Year 5	385.44	779.01
28	Year 5: July/Aug	Year 5	334.72	1113.73
29	Year 5: Sept/Oct	Year 5	330.47	1444.20
30	Year 5: Nov/Dec	Year 5	237.00	**1681.20**
31	Year 6: Jan/Feb	Year 6	303.78	303.78
32	Year 6: March/Apr	Year 6	263.75	567.53
33	Year 6: May/June	Year 6	207.08	774.61
34	Year 6: July/Aug	Year 6	304.83	1079.44
35	Year 6: Sept/Oct	Year 6	305.67	1385.11
36	Year 6: Nov/Dec	Year 6	197.65	**1582.76**
37	Year 7: Jan/Feb	Year 7	470.02	470.02
38	Year 7: March/Apr	Year 7	217.36	687.38
39	Year 7: May/June	Year 7	217.08	904.46
40	Year 7: July/Aug	Year 7	541.01	1445.47
41	Year 7: Sept/Oct	Year 7	423.17	1868.64
42	Year 7: Nov/Dec	Year 7	256.06	**2124.70**
43	Year 8: Jan/Feb	Year 8	309.40	309.40
44	Year 8: March/Apr	Year 8	254.91	564.31
45	Year 8: May/June	Year 8	290.98	855.29
46	Year 8: July/Aug	Year 8	370.74	1226.03
47	Year 8: Sept/Oct	Year 8	329.72	1555.75
48	Year 8: Nov/Dec	Year 8	229.05	**1784.80**

- To summarise the data, "subsetting IF statement" is required:

```
*p_electricity.sas;
/*Step 4: Summary of accumulating the total cost by years */
data Summ_electricitycyr(keep=period total_cost);
   set elect.electricitycyr;
   by period;
   if First.period then total_cost=0;
   total_cost+cost;
   if Last.period;
run;


proc print data=Summ_electricitycyr;
run;
```

The output:

Obs	Period	total_ cost
1	Year 1	1591.20
2	Year 2	1688.24
3	Year 3	1462.92
4	Year 4	1603.24
5	Year 5	1681.20
6	Year 6	1582.76
7	Year 7	2124.70
8	Year 8	1784.80

Ex. 8.4: Summarize total electricity cost of the electricity data set ("D:\SAS_DATASETS\ ELECTRICITY.SAS7BDAT") using the numeric variable "YEAR" with the following output:

year	total_ cost
1	1591.20
2	1688.24
3	1462.92
4	1603.24
5	1681.20
6	1582.76
7	2124.70
8	1784.80

Answers:

- Multiples BY-variables
  - When data are sorted by two BY-variables. The first variable in the BY-variables list is called **primary variable** and the remaining variable(s) are called **secondary variables**.
  - LAST.BY-variable=1 for the primary variable forces LAST.BY-variable=1 for the secondary variable(s).
  - The following statements might be required for the summarizing data SAS programs:
    - IF-THEN statement
    - IF-THEN-ELSE statement
    - DO group
    - Subsetting IF statement

- Electricity data revisit:
  - Suppose the first half year data of Year 2 are deleted. The following table shows the FIRST. and LAST.value in the PDV:

Time_Period	Period	Half_Year	FIRST .Period	LAST. Period	FIRST. Half_Year	LAST. Half_Year	Cost
Year 1: Jan/Feb	Year 1	1st half year	**1**	0	1	0	295.33
Year 1: March/Apr	Year 1	1st half year	0	0	0	0	230.08
Year 1: May/June	Year 1	1st half year	0	0	0	1	213.43
Year 1: July/Aug	Year 1	2nd half year	0	0	1	0	338.16
Year 1: Sept/Oct	Year 1	2nd half year	0	0	0	0	299.76
Year 1: Nov/Dec	Year 1	2nd half year	0	**1**	**0***	**1***	214.44
Year 2: July/Aug	***Year 2***	***2nd half year***	***1***	***0***	***1***	***0***	***219.72***
Year 2: Sept/Oct	***Year 2***	***2nd half year***	***0***	***0***	***0***	***0***	***256.59***
Year 2: Nov/Dec	***Year 2***	***2nd half year***	***0***	***1***	***0***	***1***	***276.13***
Year 3: Jan/Feb	Year 3	1st half year	1	0	1	0	321.94
Year 3: …	Year 3	…	…	…	…	…	...
Year 3: Nov/Dec	Year 3	2nd half year	0	1	0	1	183.84
…	…	…	…	…	…	…	…
Year 8: Jan/Feb	Year 8	1st half year	1	0	1	0	309.4
Year 8: …	Year 8	….	…	…	…	…	...
Year 8: Nov/Dec	Year 8	2nd half year	0	1	0	1	229.05

*LAST.Period for the observations of "Year 1" and "2nd half year" is 1. This forces LAST.Half_year to 1 whenever the "Half_Year" variable of the next observation is unchanged.

  - The following SAS program shows how to create multiple BY- variables and how to delete the part of the data using IF-THEN statement:

```
*p8_electricity.sas;
/* multiple by-variables*/
data Elect.electricityCYr2;
length Period $8 Half_Year $13;
  set Elect.electricity;
  Period=Substr(Time_period,1,6);
  if UPCASE(Substr(Time_period,9,3)) eq 'JAN' or
     UPCASE(Substr(Time_period,9,3)) eq 'MAR' or
     UPCASE(Substr(Time_period,9,3)) eq 'MAY'
 then Half_Year='1st half year';
 else Half_Year='2nd half year';
 /* to delete the year 2: Jan to Jun data*/
 if Period='Year 2' and Half_Year='1st half year' then delete;
 run;

proc print data=Elect.electricityCYr2;
run;

proc sort data=elect.electricityCYr2;
 by period half_year;
run;
```

  - The following SAS program is used to calculate the number of months is observed in each half year and the total electricity cost for each half year:

```
*p8_electricity.sas - Half year total cost calculation;
data Elect.Summ_electricityCYr2
             (keep=period half_year Total_cost Num_month);
   set Elect.electricityCYr2;
   by period half_year;
   if First.half_year then do;
                           Total_cost=0;
                           Num_month=0;
                           end;
       Total_cost+Cost;
       Num_month+2;
       if Last.half_year;
run;

proc print data=Elect.Summ_electricityCYr2;
run;
```

Ex. 8.5: Write down an equivalent SAS program code using SUM function for the SUM statement "Num_month+2" in the above program.