PART 1

20 BASICS AND BUILDING BLOCKS

Problems appear in Part 1 because

1. they're easier, or
2. they serve as building blocks for more complex problems in Part 2.

Working on these problems lets you determine whether or not you are ready to attempt the more complex problems that follow. If you miss some of the answers to these problems, study the tools and programming techniques until you are comfortable with them. In that way, you'll gain many of the skills needed to attempt the more difficult problems.

As a rule, these are intermediate (not introductory) problems. The intent of this book is to build on the skills of programmers who are already using SAS software.

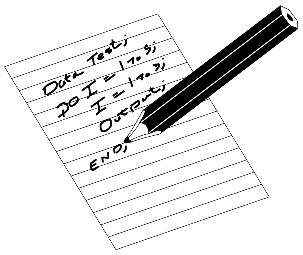
Each problem lists related tools and concepts. An appendix cross-references topics, so you can easily find other problems that use similar tools. To a small extent, the list of topics can guide you toward a solution. If you would rather not have any hints, cover up the bottom of the page as you read the problem. The problem titles are merely titles, not hints.

PROBLEM 1

Functionally Literate

What is the final value of PACKAGE in this program?

```
DATA TEST;
PACKAGE = ‘SAS’;
PART1 = SUBSTR(PACKAGE,1,1);
PART2 = SUBSTR(PACKAGE,2,1);
PART3 = SUBSTR(PACKAGE,3,1);
PACKAGE = PART3 || PART2 || PART1;
```



Tools and concepts:

- Functions
- Lengths of variables

```
DATA TEST;
PACKAGE = 'SAS';
PART1 = SUBSTR(PACKAGE,1,1);
PART2 = SUBSTR(PACKAGE,2,1);
PART3 = SUBSTR(PACKAGE,3,1);
PACKAGE = PART3 || PART2 || PART1;
```

PACKAGE has a length of 3.

The other variables, PART1-PART3, also have a length of 3. When the SUBSTR function creates a new variable, the length assigned to that variable is the same as the length of the incoming character string.

PART1, therefore, is an "S" followed by two blanks.

PART2 is an "A" followed by two blanks.

PART3 is an "S" followed by two blanks.

The fully concatenated string is nine characters long:

```
S  A  S
```

including two trailing blanks. When this string is assigned as the value of PACKAGE, there is only room to store the first three characters. Therefore, PACKAGE is an "S" followed by two blanks.

PROBLEM 2

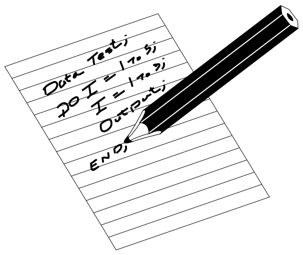
First Obs Impressions

If OLD contains 100 observations, which of those get printed?

```
OPTIONS FIRSTOBS=5 OBS=20;

PROC SORT DATA=OLD OUT=NEW;
BY ID;

PROC PRINT DATA=NEW;
```



Tools and concepts:

Global options

```
OPTIONS FIRSTOBS=5 OBS=20;

PROC SORT DATA=OLD OUT=NEW;
BY ID;

PROC PRINT DATA=NEW;
```

Both options (FIRSTOBS and OBS) remain in effect throughout the program, affecting any subsequent step that reads data. In this case, both PROC SORT and PROC PRINT are affected.

The FIRSTOBS option says that when you are reading from any source of data, begin with the 5th observation. The OBS option says when reading from any source of data, end with the 20th observation.

PROC SORT reads in observations, sorts them, and then writes them back out. Thus the effect of the options is that PROC SORT reads in only 16 observations, the 5th through 20th from OLD. Those 16 observations, in sorted order by ID, are written back out to NEW.

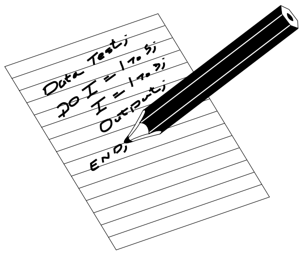
PROC PRINT then reads in the remaining observations, and prints them. Because of the FIRSTOBS option, PROC PRINT begins printing with the 5th observation from NEW. Thus, the printed observations are: out of observations 5 through 20 from OLD, those with the 12 highest values for ID.

PROBLEM 3

Keep Your Day Job

The author of these words quickly abandoned his dreams of becoming a writer and became a successful programmer instead. Under what conditions, if any, would these SAS statements execute without error?

```
DATA MEANS NOTHING; SET YOUR MIND TO THE TASK; PUT YOUR
SHOULDER TO THE WHEEL.  WITH THE PROPER RATIO OF
WORK/DESIRE –– PLUS A LITTLE LUCK –– ALL YOUR DREAMS
CAN COME TRUE.  ;
```



Tools and concepts:

Customized reporting
Variable lists

```
DATA MEANS NOTHING; SET YOUR MIND TO THE TASK; PUT YOUR
SHOULDER TO THE WHEEL.  WITH THE PROPER RATIO OF
WORK/DESIRE -- PLUS A LITTLE LUCK -- ALL YOUR DREAMS
CAN COME TRUE.  ;
```

Here are the conditions:

- The following formats must exist: WHEEL and TRUE.
- The following data sets must exist: YOUR, MIND, TO, THE, and TASK.
- The variables named in the PUT statement may or may not be contained in these data sets, subject to three restrictions:
  1. The variables THE and COME, if they exist, must be numeric.
  2. If either DESIRE or PLUS exists, both must exist.
  3. If either LUCK or ALL exists, both must exist.

For the last two pairs of variables, the order is important. The double hyphen (--) means "all variables beginning with the first and ending with the second," so the words:

```
DESIRE -- PLUS
```

denote a list of all variables beginning with DESIRE and ending with PLUS. The variables do not have to come from the same data set. However, the internal storage position of DESIRE must come before the internal storage position of PLUS for the double hyphen to be valid. Internal storage position is the same as the order in which the variables were defined. PROC CONTENTS (or the CONTENTS statement within PROC DATASETS) reveals the internal storage position as well as much more information about each variable.

If any variable in the PUT statement does not exist, the software automatically creates it. Such variables may be uninitialized (meaning they were never assigned a value), but they will "exist" in the sense that storage space in memory is reserved to hold their values. Each new uninitialized variable will be numeric in this program.

PROBLEM 4

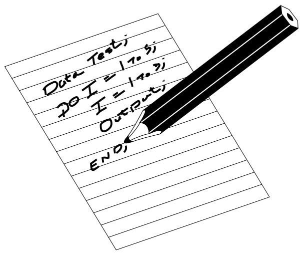
Any Comments?

This program attempts to print the results of PROC SUMMARY. Describe any errors in these statements and how to correct them.

```
*****************************;
***  COMPUTE THE AVERAGE  ***;
***  COST; PRINT RESULTS. ***;
*****************************;

PROC SUMMARY DATA=SALES;
VAR COST;

PROC PRINT;
```



Tools and concepts:

Comments
PROC MEANS/PROC SUMMARY

```
*****************************;
***  COMPUTE THE AVERAGE  ***;
***  COST; PRINT RESULTS. ***;
*****************************;

PROC SUMMARY DATA=SALES;
VAR COST;

PROC PRINT;
```

There are two problems here:

1 There is an extra semicolon within the block of comments.

2. There is no summary data set for PROC PRINT to process.

An initial asterisk comments out a SAS statement, not an entire line. Thus the semicolon after COST (in line 3) ends a comment statement. The next statement,

```
PRINT RESULTS. ***;
```

is an invalid SAS statement.

PROC SUMMARY requires an OUTPUT statement to create an output data set. Without it, PROC PRINT can't print the results of PROC SUMMARY. (It would print the most recently created SAS data set instead.)

As an alternative, add the PRINT option, eliminating PROC PRINT:

```
PROC SUMMARY DATA=SALES PRINT;
VAR COST;
```