A Reasonably Comprehensive Test of SAS® Programming Skills
Standish W. Sibley, Pacific Gas & Electric Co.

ABSTRACT
Not infrequently, a programmer is chosen for a SAS programming position after what may amount to an inappropriate interview. Perhaps congeniality is achieved and applicant and interviewer personalities mesh well, potentially resulting in a job offer for the wrong reasons. Perhaps references weren't adequately checked or honestly appraised, perhaps the applicant did not—for a variety of reasons—supply examples of code demonstrating his or her unsaid skill level. In these cases, the hiring manager has little idea of what the applicant is truly capable of, and offers of employment or salary level may be based on entirely inappropriate criteria.

This paper has associated with it a variety of test items designed to qualify an applicant, containing questions and problems revolving around real-world applications. The items range enough in difficulty to identify the beginner or the expert, and so may be used to help make the right hiring decision.

THE PROBLEM
Mistakenly hiring or promoting the wrong person for the job
Too often have I found that prospective employees and contractors who are expected to perform as knowledgeable SAS programmers cannot produce functional, maintainable and sensible SAS code of suitable quality once they are installed in the workplace. Programmer productivity may be lacking, code efficiency may suffer, even the results may be incorrect or at least questionable. And I have witnessed cases where staff morale was affected negatively by the "new kid on the block," the one who came in at too high a level, too high a salary, or too low a level of productivity, causing some amount of discontent among the other players.

Of course, in the case of contractors it is—or should be—relatively easy to terminate their relationship with your company. Employees, however, can present a somewhat more ticklish problem, especially if any probationary period has expired or it they transferred from elsewhere in the firm. I liken this situation to an old rule in managing rental properties: It's much easier to avoid getting a potentially bad tenant during the application/verification phase than it is to get rid of, one once they're in. In many cases, only a fair amount of homework and validation will ensure that the hiring manager avoids the poor prospect.

One cause: the practiced applicant
Job applicants, like rental applicants, are on their best behavior. They're as sharp and articulate as they can be. Some have been interviewed frequently and/or recently; they've improved on their mistakes and applied feedback to improving their interviewing skills. I speak from experience applying for numerous jobs and consulting opportunities. And I found that I often felt better and more experienced at interviewing than those who were interviewing me. The result of this can be that, during interviews, applicants may appear quite competent and experienced, answering general questions acceptably while impressing the interviewer with their candor and timely responses. Applicants may claim—usually unintentionally—a higher skill level than they truly possess. (It seems that fairly inexperienced programmers often think they know more than they do because they've mastered some narrow techniques or limited applications.)

Another cause—the overburdened or uncaring hiring manager
It's the hiring manager's responsibility to follow up on an applicant's reported skills, experience and qualifications. Just a few phone calls to the right people can shed a lot of light on the situation. The recommendations may be stunningly positive (then hire that applicant!) or not nearly so good. Care should be taken to contact the proper references—people who worked closely with, or supervised, the applicant, not just somebody who worked nearby and had little or no professional contact. (I recall one case where a secretary recommended the applicant for a programming position, but the applicant's former supervisor later related that she couldn't imagine how he got the job!)

The hiring manager should also make some effort to obtain samples of code the prospect has written. Is it legible? Is it purpose fairly clear? What experience level is indicated? Are any programming standards apparent and was any attempt made toward some kind of style? Is the code well-commented, or commented in a cursory fashion, or not commented at all? Is there any evidence that the job applicant actually wrote it, e.g., could s/he explain how it works or duplicate portions of it on the spot?

Assessing someone's suitability for a position is a tough proposition, making it necessary to filter out the overzealous, cut through the fluff, determine the level of aptitude, and weigh any apparent or sensed personality characteristics to separate the wheat from the chaff. The final decision could affect team morale, group productivity, and product quality and turnaround time literally for years to come. It just takes diligent work, and sometimes a little luck, to make the right decision.

EVALUATING SAS PROGRAMMING SKILLS BY ASKING THE RIGHT TECHNICAL QUESTIONS
In conjunction with these suggested evaluations, one solution to the hiring dilemma is a comprehensive skill test designed to identify an applicant's SAS programming abilities. Such a test should cover a broad range of skill levels from those of near neophytes to those of macro mavens, and should at least summarily test proficiency in a variety of SAS products, e.g., the macro language, SAS/AF, SAS/Adj• the DATA step, at least a few common PROCs, DMS, etc.

Suggested test items are presented below, grouped according to the SAS products whose proficiency they gauge. Some overlap of areas is inevitable, e.g., one generally needs to know the base product before escalating to the domain of the macro language. Some of the areas may cover a range of abilities, requiring knowledge in multiple areas to complete successfully. No attempt was made to standardize these items, and there is no scoring method provided. Nevertheless, different levels of difficulty will be apparent, and the overall goal—judging a job applicant's technical abilities—will be facilitated by these test items and others like them. If particular areas or applications are neglected or should be stressed in a given hiring situation, that need may be met by writing your own items. These may include the realms of intermediate/advanced statistics, report-writing skills, an emphasis on Release 6 or on operating systems you use, or on other SAS products.

A set of suggested answers is presented at the end. It must be borne in mind that there's more than one way to skin a cat, especially with

1 It should be noted that there may be legal ramifications to basing a hiring decision on a non-standardized test; consult your legal advisor for details.
SAS. An attempt was made to suggest different ways of reaching the same goal, each of which works, but each of which reflects a different level of skill, efficiency, or simplicity. The solutions may not be comprehensive and they may not apply to some operating systems—IBM operating systems are stressed here. The code provided has been tested on IBM’s OS/VS using both Version 5 and Release 6 of SAS, but entirely different code may accomplish the stated goals, and may still give a usable indication of an applicant’s abilities.

Permission to duplicate and use this paper and the test items contained herein is granted freely, upon the condition that the author be credited.

THE TEST

Some of the test items make reference to SAS data sets named MASTER.FEB, MASTER.MAR, and MASTER.APR. Assume that each of these data sets has the following contents:

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DATA TYPE</th>
<th>LENGTH</th>
<th>LABEL</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION</td>
<td>CHARACTER</td>
<td>20</td>
<td>REGION NAME</td>
<td></td>
</tr>
<tr>
<td>DIVISION</td>
<td>CHARACTER</td>
<td>2</td>
<td>DIVISION CODE</td>
<td></td>
</tr>
<tr>
<td>SALES</td>
<td>NUMERIC</td>
<td>5</td>
<td>TOTAL SALES</td>
<td>DOLLAR12.</td>
</tr>
<tr>
<td>DATE</td>
<td>NUMERIC</td>
<td>3</td>
<td>TRANSACTION DATE</td>
<td>MONDAY.</td>
</tr>
<tr>
<td>TAYRYPE</td>
<td>NUMERIC</td>
<td>2</td>
<td>TRANSACTION TYPE</td>
<td></td>
</tr>
<tr>
<td>ACCT</td>
<td>CHARACTER</td>
<td>8</td>
<td>CUSTOMER ACCOUNT NO.</td>
<td></td>
</tr>
</tbody>
</table>

SAS/Base Test Items:

1. You have access to a data set, MASTER.FEB. Write a brief program to report the number of observations in the data set. Try to use a header (HEADER=} option on the FILE statement) to print our company name and the date the report was run. Your report is to have the roughly the following layout:

   XYZ COMPANY
   19FEB91
   SAS DATA SET MASTER.FEB HAS n, mm OBSERVATIONS.

2. Using the same data set (MASTER.FEB), invoke PROC SUMMARY to calculate the mean sales per region. Report the results, sorted by region, with PROC PRINT. Double-space the report.

3. You have access to a SAS format, $DIV, which relates the division code (DIVISION) to its 20-character name. ($DIV does not have an "OTHER" value.) Write a DATA step to read MASTER.FEB and create a new variable containing the division name. Using a SUM statement, count the number of observations for which you cannot successfully obtain a division name from the format, and output the count to the corresponding ACCT in a macro.

4. Data sets MASTER.FEB, MASTER.MAR, and MASTER.APR are sorted by ascending ACCT. Each contains only one observation per ACCT.

   Write a DATA step to merge MASTER.FEB, MASTER.MAR, and MASTER.APR by ACCT, outputting only those observations for account numbers (ACCT) found in all three data sets.

5. Data set MASTER.FEB is sorted by ascending ACCT and ascending DATE within ACCT.

   Some occurrences of ACCT contain only null values (hexadecimal '00's). Create a data set containing only the chronologically latest observation for each account, discarding ACCTs which contain only null values.

6. PERCENT is an integer numeric variable, the values for which range from 0 to 100. What length (in bytes) is the minimum necessary to store PERCENT? Please write the corresponding LENGTH statement.

7. Raw input data contains a PAYMENT field, coded like "$234,567.89" right-justified in columns 1-11. How would you read such a value into a numeric SAS variable? Please write any code necessary to do this.

8. What does an ATTRIB statement accomplish? Write an ATTRIB statement.

9. How can one keep the value of an "in variable" from a SET or MERGE or UPDATE on the resulting output data set?

10. What does "NOTE: MERGE STATEMENT HAS MORE THAN ONE DATA SET WITH REPEATES OF BY VALUES" mean?

11. When must a RUN statement be used? What does the RUN statement accomplish?

12. Where can "compiled" or "permanent" SAS formats (formats which have been processed successfully by PROC FORMAT) be stored? How can you then display the values in a stored permanent or compiled format?

13. Name three ways to submit a command to TSO (or CMS or DOS) from within a SAS program. How may you then check the the return code from the operating system?

14. What is the significance of the "OF" in "TOTAL = SUM (OF PAYMENT3-PAYMENT5, PAYMENT7, PAYMENT9);"?

15. Explain, generally, the differences between:

   "ARRAY QTR (* JAN FEB MAR);"
   "ARRAY QTR1 (* JAN FEB MAR);"

16. Approximately what date is represented by the SAS date value of "7301"?

SAS Macro Facility Items:

17. Write a DATA step which reads data set MASTER.FEB and places the value for REGION from the second observation into a macro variable also called REGION.

18. When must you use %QUOTE ("blind quote") instead of %QUOTE or %STR to quote a macro variable value? How can you quote a macro variable value which contains NOT, OR, or "=" (a minus sign)?

19. What SAS system options might you manipulate (set on or off) when you are debugging macro code? What two macro debugging system options are incompatible with each other (i.e., if one is on, it overrides the other)?

20. Explain how the "source macro autocall" (or just "autocall") facility works.

21. What method of commenting code works in the base product as well as in macro code? What commenting method is unique to the

1602
macro facility? Name two other ways to comment base SAS code (not macro code).

22. Let's say you have autocalled a macro which has serious errors in it, such as the member name not being the same as the macro name. List two ways to get SAS to use "(compile)" a corrected version of the macro source code.

Display Manager System (DMS) Items.

23. How can one clear a screen in DMS? How can one recall the section of code which was last submitted for execution?

24. What is the fastest way to place the cursor on the command line of the PROGRAM or PGM screen?

25. How can you view or change the commands associated with the function keys? Why might SAS not save your edited commands from session to session?

SAS/AF Items (Version 5):

26. Name three types of AF screens.

27. How can you be sure that function keys in AF will behave the same for all users of an AF system, i.e., a user doesn't have AF keys set up that will do something unexpected?

28. What values may _DCALL have?

29. What is the simplest way to conditionally display a message on a program screen?

30. Where are AF screens and programs stored?

SAS/FSP Items (Version 5):

31. How does one get into the Screen Modification Menu to define or revise an FSEDIT/FSBROWSE screen?

32. An application's FSEDIT/FSBROWSE screen displays a variable called CITY. What command or commands would find the next observation with the value of ROANOKE for city?

33. In Field Identiﬁcation phase, how does one indicate to FSP that a variable is not to be displayed on a screen?

34. A speciﬁcation calls for an FSEDIT/FSBROWSE screen to accept a date in MMDDYY format, store it as a SAS date value, and display it in MMDDY Y format. How may this be accomplished?

35. An application's FSEDIT/FSBROWSE screen displays a variable labeled "Last inquiry date". How may a user determine what SAS variable is being displayed with this label, as well as its type, FORMAT used and INFORMAT used?

SOLUTIONS

1. A. Obtain the information.

   Poor example:
   ```sas
   DATA REPORT;
   SET MASTER.FEB END=EOF;
   RETAIN NUMOBS 0;
   NUMOBS = NUMOBS + 1;
   IF EOF THEN
     OUTPUT; * OR JUST "IF EOF".*;
   RUN;
   ```

   Better example (does not work with SAS tape-format data sets):
   ```sas
   DATA REPORT (KEEP = NUMOBS);
   SET MASTER.FEB POINT= N NOBS=NOBS;
   NOBS = NOBS + 1; * NOBS CANNOT BE KEPT*;
   OUTPUT;
   STOP; * WITHOUT "STOP," *;
   ```

2. Report the results.

   OPTION DQUOTE;
   DATA NULL;
   SET REPORT;
   FILE PRINT HEADER=HEADER;
   PUT 'SAS DATA SET MASTER.FEB HAS ';
   RETURN;
   HEADER:
   PUT 'XYZ COMPANY' @70 H,SYSDATE";
   RETURN;
   RUN;

3. Code:

   ```sas
   PROC SUMMARY DATA=MASTER.FEB (KEEP = SALES REGION);
   CLASS REGION;
   VAR SALES;
   OUTPUT OUT=SALES {DROP SUM=REGSALES;
   RUN;
   ```

   *NO PROC SORT NECESSARY--PROC SUMMARY OUTPUT*;
   *IS ALREADY SORTED BY THE CLASS VARIABLE(S).*;
   ```sas
   PROC PRINT DOUBLE DATA=SALES;
   VAR REGION REGSALES;
   FORMAT REGSALES DOLLAR11.; * OPTIONAL *;
   RUN;
   ```

3. Code:

   ```sas
   DATA UNIVERSE SAMPLE FAILED (KEEP = NFAILED);
   SET MASTER.FEB (KEEP = DIVISION) END=EOF;
   DIVNAME = PUT (DIVISION,DIVISION);
   IF DIVNAME EQ DIVISION THEN
     NFAILED + 1;
   OUTPUT UNIVERSE;
   IF MOD ( N ,10) EQ 0 THEN
     OUTPUT SAMPLE;
   **"IF INT( N /10 )*10 EQ _N_ " WORKS TOO *;
   IF EOF THEN
     OUTPUT FAILED;
   ```

   RUN;

1603
4. Code:
DATA ALL;
MERGE MASTER.FEB (IN=IN_FEB)
MASTER.MAR (IN=IN-MAR)
MASTER.APR (IN=IN-APR);
BY Acct;
IF IN_FEB AND IN_MAR AND IN_APR;
RUN;

5. Code:
DATA;
SET MASTER.FEB;
BY Acct DATE;
IF Acct EO 'QOOOOOODOOOOOOO'O THEN DELETE;
IF LAST.DATE;
RUN;

6. Code:
LENGTH PERCENT 2;

7. One acceptable way:
INPUT Temp $1-11;
PAYMENT = INPUT (COMPRESS (TRANSLATE (Temp, '$', '11.));
DROP Temp;
A better way:
INPUT PAYMENT Commal!.;

8. An ATTRIB statement defines a new variable and, optionally, its LENGTH, FORMAT, INFORMAT and LABEL attributes.

9. One cannot keep an in-variable directly; however, one can assign the value of an in-variable to a variable which can be kept. An example:
DATA;
SET PART1 (IN = IN_PART1)
PART2 (IN = IN_PART2);
IN_1 = IN_PART1;
IN_2 = IN_PART2;

10. It means that at least two of the data sets being merged have duplicated values of the by-variables. This may cause unintended results in the output data set, even though SAS follows certain rules in this instance.

11. In interactive SAS, the RUN statement should be used to terminate a DATA or PROC step that has been entered and needs to be executed. As an alternative, one may begin a subsequent DATA or PROC step, but this can be confusing and error-inducing.

12. MVS: In a PDS called a format library.
CMS: In a file with the filetype of TEXT, or in a TXTLIB.
DOS: In a DOS file referenced by a LIBNAME statement. Compiled (permanent) formats may be viewed with PROC FMTLIB (Release 5) or with the PROC FORMAT CNTLOUT= operand (Release 6).

13. %CMS cmscommand; %TSO tsocommand; %TSO tsocommand;
CMS:CMS cmscommand; %CMS cmscommand;

14. The "OF" implies that variables PAYMENT3, PAYMENT4 and PAYMENT5 should be summed as one addend. Without the "OF," the difference of PAYMENT3 and PAYMENTS (PAYMENT3 minus PAYMENTS) would be used as the addend.

15. The first array is an "implicit array," subscripted by the automatic variable _L_. The second is "explicit," and is subscripted by the numeric variable _L_. The third is explicit also, but may be subscripted by any numeric variable.

16. December 31, 1961. (730 days from January 1, 1960. 1960 was a leap year, containing 366 days.)

17. Code:
One way:
DATA NULL;
SET MASTER.FEB;
IF N EQ 2 THEN
CALL SYMPUT ('REGION', REGION):
RUN;
A better way:
DATA NULL;
SET MASTER.FEB (FIRSTOBS=2 OBS=2):
CALL SYMPUT ('REGION', REGION):
RUN;

18. You should use %BQUOTE when the resolved macro variable value contains either a percent sign (%) or an ampersand (&).
%NRBQUOTE ("No Rescan Blind QUOTE") can quote resolved values containing AND, OR, or ".%NRQUOTE and %SUPERQ may also be suitable.

19. MPRINT, MACROGEN (MGEN), SYMBOLGEN (SGEN) and MLOGIC are the most likely candidates. Others, like MACRO, MAUTOSOURCE, MRECALL, MCOMPILE, MERROR, MSIZE, MSYMSIZE, and MWORK are changed less frequently. MLOGIC overrides MPRINT.

20. When a macro is invoked (and option MAUTOSOURCE is in effect), SAS checks to see if a macro with the same name as the one being invoked has been compiled. If not, SAS attempts to include the macro source code from the library referenced by the DDNAME=FILEDEF=LIBREF SASAUTOS. In either case, the macro, if found, is executed.

21. /* comment */ works in both places.
"%" comment; is unique to the macro facility.

"*/ comment;" and "COMMENT comment;" work in the base product. Under MVS, "*/" in columns 1 and 2 of the source code effectively comment out the rest of the code. (MVS interprets this as an end-of-file marker, a holdover from punched cards.)
22. Set option MRECALL on, then invoke the macro again, or INCLUDE or %INCLUDE the member containing the source code and SUBMIT it.

23. One may use the CLEAR line command (or CLEAR function key, if one is defined). The RECALL command or function key will restore previously-submitted code.

24. In Version 5, exit the OUTPUT screen if necessary. Press HOME and TAB. In Release 6, use the PGM/PROGRAM command or function key. (From the SAS LOG window, the NEXT command or function key will work.)

25. The KEYS command or function key will bring up a screen of function keys and their commands. The commands may then be edited. If you do not have a SASUSER PROFILE allocated, revisions will not be saved.

26. Choose from among: MENU, PROGRAM, HELP, FORMS, KEYS, CBT, and maybe others.

27. Create a screen called name.KEYS. Using PROC BUILD, edit the desired key values into it, then rename it to DISPLAY.KEYS or BUILD.KEYS.

28. INIT, PARSE, END, or CANCEL.

29. Assign the message to the macro variable _DMSG.

30. In an AF catalog, software may be stored in a macro library or with the screens themselves.

31. One issues the MOD or MODIFY command from the command line. If the screen is password-protected, the password is also required at this point.

32. FIND CITY = ROANOKE. Alternatively, NAME CITY, then LOCATE ROANOKE. Alternatively, STRING CITY and SEARCH ROANOKE. All of these commands may be abbreviated.

33. One presses the HOME function key (not the HOME key). By default, this is PF12. Or one may enter HOME on the command line.

34. The DATA step must first associate the date variable with the MMDDYY format and with the MMDDYV informal (MMDDYV defaults to MMDDYVB.)

35. The user can place the cursor on the field, then press the HELP function key (PF1 by default). Entering HELP on the command line, the cursor on the field, and pressing ENTER will accomplish the same thing.

SAS, SAS/AF, and SAS/FSP are registered trademarks of SAS Institute, Inc. Cary, N.C.