SURVIVING THE SAS® MACRO JUNGLE BY USING YOUR OWN PROGRAMMING TOOLKIT

KEVIN RUSSELL
TOOLS FOR YOUR MACRO PROGRAMMING TOOLKIT

• The DOSUBL function
• The stored compiled macro facility
• Macro debugging system options
• The CALL SYMPUTX routine
• Using comments within a macro definition
TOOLS FOR YOUR MACRO PROGRAMMING TOOLKIT (continued)

• The %PUT statement
• The MFILE system option
• Read-only macro variables
• The macro IN operator
• SASHELP.VMACRO
DOSUBL FUNCTION: OVERVIEW

Provides you with the ability to move macro variables to and from the calling environment

Syntax
DOSUBL( x)
DOSUBL FUNCTION: ABILITIES

Like CALL EXECUTE, DOSUBL gives you the ability to do two things:

• conditionally execute a macro based on DATA step logic
• pass in a data set variable’s value as the value for a macro parameter

DOSUBL gives you the added ability to do the following:

• import macro variables from and then export macro variables to the calling environment
DOSUBL FUNCTION: EXAMPLE 1 CODE

%macro tryit(x);
%global mymac;
data _null_;  
  if &x=1 then mymac=2;  
  else mymac=3;  
  call symputx('mymac', mymac);  
run;
%mend;

%let mymac=4;
data _null_;  
  rc=dosubl('%tryit(1)');  
  x=symget('mymac');  
  put x= '(should be 2)';  
run;
DOSUBL FUNCTION: EXAMPLE 1 LOG OUTPUT

687  %let mymac=4;
688  data _null_
689     rc=dosubl('%tryit(1)');
690     x=symget('mymac');
691     put x= '(should be 2)';
692  run;
MLOGIC(TRYIT):  ...  %GLOBAL  MYMAC                                                                          MPRINT(TRYIT):   data _null_
SYMBOLGEN:  Macro variable X resolves to 1                                                              MPRINT(TRYIT):   ...
NOTE: DATA statement used (Total process time):
         real time    0.01 seconds  cpu time     0.00 seconds
MLOGIC(TRYIT):  Ending execution.
x=2 (should be 2)
DOSUBL FUNCTION  (continued)

if first.id then do;
  %tryit(x)
end;

if first.id then do;
  rc=dosubl(cats('%tryit(','x',')));
end;
**DOSUBL FUNCTION: EXAMPLE 2 CODE**

```sas
%macro mymac(hosp_id);
  title "Data for hospital: &hosp_id";
  proc report data=hospitals(where=(hospital_id="&hosp_id"));
    column hospital_id quarter number_of_patients;
    define hospital_id / order;
  run;
%mend;

data _null_;  
  set hospitals;  
  by hospital_id;  
  if first.hospital_id then do;  
    rc=dosubl(cats('%mymac(',hospital_id,')'));  
  end;  
  run;
```

Macro invocation passed to DOSUBL:

```
%mymac(A100)
```
**DOSUBL FUNCTION: EXAMPLE 2 OUTPUT**

<table>
<thead>
<tr>
<th>hospital_id</th>
<th>quarter</th>
<th>number_of_patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>A100</td>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hospital_id</th>
<th>quarter</th>
<th>number_of_patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>A200</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>195</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>190</td>
</tr>
</tbody>
</table>
STORED COMPILED MACRO FACILITY: OVERVIEW

- Compiles macros only once, which saves time with later executions
- Saves compiled macros in a library that you specify
- Protects your source code
STORED COMPILED MACRO FACILITY: OVERVIEW (continued)

• The compiled macro is stored in an unreadable machine language
• The macro cannot be decompiled after it is compiled
• %MACRO statement option, SECURE
options mprint mlogic;

%macro test;
%if %sysfunc(exist(work.new))=0
%then %do;
data new;
   set sashelp.cars;
run;
proc freq data=new;
   tables make*model / out=outfreq;
run;
proc print data=outfreq;
run;
%end;
%mend;
%test

#pharmasug
STORED COMPILED MACRO FACILITY: EXAMPLE 1 LOG OUTPUT

MLOGIC(TEST):  Beginning execution.

MLOGIC(TEST):  %IF condition
%sysfunc(exist(work.new))=0 is TRUE

MPRINT(TEST):  data new;
MPRINT(TEST):  set sashelp.cars;
MPRINT(TEST):  run;

NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: The data set WORK.NEW has 428 observations and 15 variables.
NOTE: DATA statement used (Total process time):
    real time           0.12 seconds
    cpu time            0.04 seconds

MPRINT(TEST):  proc freq data=new;
MPRINT(TEST):    tables make*model / out=outfreq;
MPRINT(TEST):  run;

NOTE: There were 425 observations read from the data set WORK.OUTFREQ.
NOTE: PROCEDURE FREQ used (Total process time):
    real time           11.24 seconds
    cpu time            10.96 seconds

MPRINT(TEST):  proc print data=outfreq;
MPRINT(TEST):  run;

NOTE: There were 425 observations read from the data set WORK.OUTFREQ.
NOTE: PROCEDURE PRINT used (Total process time):
    real time           0.28 seconds
    cpu time            0.28 seconds

MLOGIC(TEST):  Ending execution.
STORED COMPILED MACRO FACILITY: EXAMPLE 2 CODE

libname mylib 'c:\dstep\compiled';

options mstored sasmstore=mylib;

%macro mytest/store secure;
STORED COMPILED MACRO FACILITY: EXAMPLE 3

Code:  
%mytest

Log Output:  
NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: The data set WORK.NEW has 428 observations and 15 variables.
NOTE: DATA statement used (Total process time):
   real time          0.04 seconds
   cpu time           0.01 seconds
NOTE: There were 428 observations read from the data set WORK.NEW.
NOTE: The data set WORK.OUTFREQ has 425 observations and 4 variables.
NOTE: PROCEDURE FREQ used (Total process time):
   real time          10.10 seconds
   cpu time           10.06 seconds
NOTE: There were 425 observations read from the data set WORK.OUTFREQ.
NOTE: PROCEDURE PRINT used (Total process time):
   real time          0.21 seconds
   cpu time           0.21 seconds
MACRO DEBUGGING SYSTEM OPTIONS: OVERVIEW

- SYMBOLGEN
- MPRINT
- MLOGIC
MACRO DEBUGGING SYSTEM OPTIONS: SYMBOLGEN

Writes out the resolved value of a macro variable each time it is referenced

(in the SAS log)

SYMBOLGEN: Macro variable X resolves to 1
MACRO DEBUGGING SYSTEM OPTIONS: MPRINT

Writes out all the non-macro SAS statements generated by macro execution

(in the SAS log)

MPRINT(TEST): proc print data=sashelp.cars;
MPRINT(TEST): run;
MACRO DEBUGGING SYSTEM OPTIONS: MLOGIC

Writes out important information about the macro’s execution

718  %macro test(x);
719  %let newx=%eval(&x+1);
720  %if &newx<10 %then %put Hello;
721  %mend;
722
723  options mlogic;
724  %test(5)

MLOGIC(TEST):  Beginning execution.
MLOGIC(TEST):  Parameter X has value 5
MLOGIC(TEST):  %LET (variable name is NEWX)
MLOGIC(TEST):  %IF condition &newx<10 is TRUE
MLOGIC(TEST):  %PUT Hello
Hello
MLOGIC(TEST):  Ending execution.
MACRO DEBUGGING SYSTEM OPTIONS: EXAMPLE CODE

Options mautosource sasautos=('c:\dstep') nosymbolgen nomprint nomlogic;

%macro test;
  %keepnums(sashelp.cars)

%mend;
%test
%test
MACRO DEBUGGING SYSTEM OPTIONS: EXAMPLE LOG OUTPUT

options mautosource sasautos=('c:\dstep') NOsymbolgen NOmprint NOmlogic ;
58
59  %macro test;
60  %keepnums(sashelp.cars)
61
62  %mend;
63  %test
NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: DATA statement used (Total process time):
    real time           0.73 seconds
    cpu time            0.73 seconds
ERROR: "MSRPINVOICEENGINE_SIZECYLINDERSHORSEPOWERMPG_CITYMPG_HIGHWAYWEIGHTWHEELBASELENGTH"
contains more than 32 characters.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE PRINT used (Total process time):
    real time           0.00 seconds
    cpu time            0.00 seconds
MACRO DEBUGGING SYSTEM OPTIONS: EXAMPLE LOG OUTPUT

MLOGIC(TEST):  Beginning execution.
MLOGIC(KEEPNUMS):  Beginning execution.
MLOGIC(KEEPNUMS):  This macro was compiled from the autocall file c:\dstep\keepnums.sas
MLOGIC(KEEPNUMS):  Parameter DSN has value sashelp.cars
MPRINT(KEEPNUMS):  data _null;
SYMBOLGEN:  Macro variable DSN resolves to sashelp.cars
MPRINT(KEEPNUMS):  set sashelp.cars;
MPRINT(KEEPNUMS):  length numnames $ 32767 name $ 32;
MPRINT(KEEPNUMS):  array nums(*) _numeric_;
MPRINT(KEEPNUMS):  do i=1 to dim(nums);
MPRINT(KEEPNUMS):  call vname(nums(i),name);
MPRINT(KEEPNUMS):  if name ne 'I' then
numnames=cats(numnames,name);
MPRINT(KEEPNUMS):  call symputx('numnames',numnames);
MPRINT(KEEPNUMS):  end;
MPRINT(KEEPNUMS):  run;

NOTE: There were 428 observations read from the data set SASHELP.CARS.

NOTE: DATA statement used (Total process time):
      real time    1.25 seconds
      cpu time    0.93 seconds
SYMBOLGEN:  Macro variable DSN resolves to sashelp.cars
MPRINT(KEEPNUMS):  proc print data=sashelp.cars;
SYMBOLGEN:  Macro variable NUMNAMES resolves to MSRPInvoiceEngineSizeCylindersHorsepowerMPG_CityMPG_HighwayWeightWheelbaseLength
ERROR:
"MSRPINVOICEENGINESIZECYLINDERSHORSEPOWERMPG_CITYMPG_HIGHWAYWEIGHTWHEELBASELENGTH"
contains more than 32 characters.
MPRINT(KEEPNUMS):  var
MPRINT(KEEPNUMS):  MSRPInvoiceEngineSizeCylindersHorsepowerMPG_CityMPG_HighwayWeightWheelbaseLength;
MPRINT(KEEPNUMS):  run;
NOTE: The SAS System stopped processing this step because of errors.
CALL SYMPUTX ROUTINE: OVERVIEW

Includes all the functionality of the CALL SYMPUT routine with a few improvements

Syntax
CALL SYMPUTX(macro-variable,value<,symbol-table>);
CALL SYMPUT ROUTINE: EXAMPLE 1 CODE

data total;
   set sales_data end=last;
   total_sales+sales;
   if last then call symput("total_sales",strip(total_sales));
run;
%put &=total_sales;
TOTAL_SALES=17240

proc print data=total;
   title "Total sales for this region:  &total_sales";
run;
CALL SYMPUTX ROUTINE: TWO MOST USEFUL IMPROVEMENTS

• Left-justifies arguments and trims any resulting trailing blanks
• Gives you the ability to specify which symbol table the macro variable is stored in
CALL SYMPUT ROUTINE: EXAMPLE 1 CODE  (continued)

data total;
  set sales_data end=last;
  total_sales+sales;
  if last then call symput("total_sales",strip(total_sales));
run;
CALL SYMPUT ROUTINE: EXAMPLE 2 CODE

data total;
  set sales_data end=last;
  total_sales+sales;
  if last then call symput("total_sales", total_sales);
run;
proc print data=total;
  title "Total sales for this region: $&total_sales";
run;

<table>
<thead>
<tr>
<th>Obs</th>
<th>region</th>
<th>sales</th>
<th>total_sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 N</td>
<td>625</td>
<td>625</td>
<td></td>
</tr>
<tr>
<td>2 N</td>
<td>887</td>
<td>1512</td>
<td></td>
</tr>
<tr>
<td>3 N</td>
<td>243</td>
<td>1755</td>
<td></td>
</tr>
<tr>
<td>4 N</td>
<td>986</td>
<td>2741</td>
<td></td>
</tr>
<tr>
<td>5 N</td>
<td>639</td>
<td>3380</td>
<td></td>
</tr>
<tr>
<td>6 N</td>
<td>510</td>
<td>3890</td>
<td></td>
</tr>
<tr>
<td>7 N</td>
<td>455</td>
<td>4345</td>
<td></td>
</tr>
<tr>
<td>8 N</td>
<td>12895</td>
<td>17240</td>
<td></td>
</tr>
</tbody>
</table>
CALL SYMPUTX ROUTINE: IF STATEMENT

if last then call symputx("total_sales", total_sales);
CALL SYMPUTX ROUTINE: EXAMPLE CALL ROUTINE

%global x;

call symputx(cats('var',_n_),x,'g');
CALL SYMPUTX ROUTINE: OPTIONAL ARGUMENTS

call symputx('x',x,'g');

G: specifies the global symbol table
L: specifies the most local symbol table that exists
CALL SYMPUTX ROUTINE: EXAMPLE 3

call syputx(cats('var',_n_),x,'g');
USING COMMENTS WITHIN A MACRO DEFINITION: Overview

Asterisk-style comments
* this is my comment;

Macro-style comments
%* this is my comment;

PL/1-style comments
/* this is my comment */
*asterisk style comment;

%macro one;
%let x=10;
data new;
* The %let x = value in the above section is used to set the lower limit;
    set one;
    if x=&x;
    run;
%mend;
%one
COMMENTS: EXAMPLE 1 CODE

*asterisk style comment;
%macro one;
%let x=10;
data new;
* The %let x = value in the above section is used to set the lower limit;
set one;
if x>=&x;
run;
%mend;
%one
COMMENTS: EXAMPLE 1 LOG

* The %let x = value in the above section is used to set the lower limit;

    377  %one
    MLOGIC(ONE):  Beginning execution.
    MPRINT(ONE):  data new;
    MLOGIC(ONE):  %LET (variable name is X)
    MPRINT(ONE):  * The set one;

NOTE: Line generated by the macro variable "X".
    377  value in the above section is used set the lower limit
        -----
        22
        76

ERROR: The right-hand operand must be an array name or a constant value list. The specified name the, is not an array.

    MPRINT(ONE):  if x>=value in the above section is used to set the lower limit;
    MPRINT(ONE):  run;
COMMENTS: EXAMPLE 2 CODE

%* macro style comment;
%macro test;
%* The rest of the developer's comments in this section discuss the metadata;
    proc contents data=sashelp.cars out=chars noprint;
    run;
%*subset the data to look at only the character variables.;
data chars;
    set chars(where=(type=2));
    run;
%mend;
%test(test)
%macro test;
%* The rest of the developer's comments in this section discuss the metadata;
proc contents data=sashelp.cars out=chars noprint;
run;
%*subset the data to look at only the character variables.;
data chars;
  set chars(where=(type=2));
run;
%mend;

%test
COMMENTS: EXAMPLE 3 CODE

/* PL/1 style comment */

/*
The following special characters can all be safely used within this type comment:
\" \" \% ;
*/
%PUT STATEMENT: OVERVIEW

Writes out the value of the macro variable and other macro variable information

Syntax

%PUT <text | _ALL_ | _AUTOMATIC_ | _GLOBAL_ | _LOCAL_ | _READONLY_ | _USER_ | _WRITABLE_> ;
%PUT STATEMENT: EXAMPLE 1

**Code:**
```sas
data new;
  set sashelp.cars;
  if make='HONDA';
run;
```

**Log Output:**
```sas
NOTE: There were 428 observations read from the data set SASHELP.CARS.
NOTE: The data set WORK.NEW has 0 observations and 15 variables.
```
```sas
2066  Data new;
2067    set sashelp.cars;
2068    put make=;
2069    if make='HONDA';
2070  run;
```

```sas
Make=Acura
Make=Acura
Make=Acura
Make=Acura
Make=Audi
Make=Audi
```
%PUT STATEMENT: EXAMPLE 1

Log Output:
2077  data total;
2078   set sales_data end=last;
2079   total_sales+sales;
2080   if last then call symputx('total_sales',total_sales);
2081   run;

NOTE: There were 8 observations read from the data set WORK.SALES_DATA.
NOTE: The data set WORK.TOTAL has 8 observations and 3 variables.
NOTE: DATA statement used (Total process time):
       real time       0.02 seconds
       cpu time       0.01 seconds

SYMBOLGEN: Macro variable TOTAL_SALES resolves to 17240
2082   %put &=total_sales;
TOTAL_SALES=17240
options nosymbolgen nomlogic;

%macro test;

%do i=1 %to 5;
  %let x=%sysevalf(%sysfunc(ranuni(100))*100,floor);
  %let y=%sysevalf(%sysfunc(ranuni(100))*100,floor);
  %put &=i;
  %put &=x;
  %put &=y;
  %put ----------------------;
%end;
%mend;
%test

#pharmasug
%PUT STATEMENT: EXAMPLE 2 LOG OUTPUT

2167
2168  %test
I=1
X=83
Y=66
----------------------
I=2
X=12
Y=91
----------------------
I=3
X=96
Y=72
----------------------
I=4
X=10
Y=24
----------------------
I=5
X=95
Y=80
----------------------
%PUT STATEMENT: ARGUMENTS

_ALL_

_AUTOMATIC_

_GLOBAL_

_LOCAL_

_READONLY_

_USER_
Shows all of the user-generated global and local macro variables.

_WRITABLE_
%PUT STATEMENT: EXAMPLE ARGUMENT

Code:
%put _user_;
MFILE SYSTEM OPTION: OVERVIEW

Enables you to route the non-macro code generated by your macro (MPRINT output) to an external file

Syntax
MFILE | NOMFILE
MFILE SYSTEM OPTION: EXAMPLE 1 LOG OUTPUT

837  data a;
838  set sashelp.cars;
839    target=input(make,8.);
840  run;

NOTE: Invalid argument to function INPUT at line 839 column 10.
MFILE SYSTEM OPTION: EXAMPLE 1 LOG OUTPUT

%macro test;
844  data a;
845   set sashelp.cars;
846   target=input(make,8.);
847  run;
848  %mend;
849  %test
MLOGIC(TEST):  Beginning execution.
MPRINT(TEST):   data a;
MPRINT(TEST):   set sashelp.cars;
MPRINT(TEST):   target=input(make,8.);
MPRINT(TEST):   run;

NOTE: Invalid argument to function INPUT at line 849 column 39.
MFILE SYSTEM OPTION: EXAMPLE 2 CODE

```sas
filename mprint 'c:\dstep\mycode.sas';
options mprint mfile;

%macro test(var);
data a;
   set sashelp.cars;
   target=input(&var,8.);
run;
%mend;
%test(make)
```

#pharmasug
Contents of MyCode.sas:

data a;
   set sashelp.cars;
   target=input(make,8.);
run;

Log Output:

data a;
  set sashelp.cars;
  target=input(make,8.);
  run;

NOTE: Invalid argument to function INPUT at line 865 column 10.
READ-ONLY MACRO VARIABLES: OVERVIEW

Syntax

\%GLOBAL / READONLY \texttt{macro-variable=value};

Or

\%LOCAL / READONLY \texttt{macro-variable=value};
READ-ONLY MACRO VARIABLES: EXAMPLE CODE

%let date=%sysfunc(intnx(month,%sysfunc(today()),0));

Set salesdata(where=(date<&date));
READ-ONLY MACRO VARIABLES: EXAMPLE PROTECTED

Code:
%global / readonly
date=%sysfunc(intnx(month,%sysfunc(today()),0));

Log Output:
ERROR: The variable DATE was previously declared as READONLY and cannot be re-declared.
MACRO IN OPERATOR: OVERVIEW

Gives you the ability to determine whether a single value is contained in a specified list of values.
MACRO IN OPERATOR: DATA STEP EXAMPLE

data new;
  set sashelp.cars;
  if make in('Acura' 'Audi' 'Lexus' 'Infiniti' 'Mercedes-Benz')
  then cartype='Luxury';
run;
/* before SAS 9.3 */
options mlogic sgen mprint;

%macro test(make);
%if %bquote(&make)=Acura or %bquote(&make)=Audi or %bquote(&make)=Lexus or %bquote(&make)=Infiniti or %bquote(&make)=%str(Mercedes-Benz) %then %do;
data new;
   set sashelp.cars(where=(make="&make"));
run;
proc print;run;
%end;
%mend;
%test(Mercedes-Benz)
/* SAS 9.3 and later */

options mlogic sgen mprint minoperator mindelimiter=' ','.';

%macro test(make);
%if %bquote(&make) in (Acura,Audi,Lexus,Infiniti,%str(Mercedes-Benz)) %then %do;
data new;
   set sashelp.cars(where=(make="&make"));
rundata new;
   set sashelp.cars(where=(make="&make"));
run;
proc print;run;
%end;
%mend;
%mend;

%test(Mercedes-Benz)
MACRO IN OPERATOR: EXAMPLE 3 CODE

options minoperator;

options minoperator mindelimiter=’,’;

%macro test(make)/minoperator mindelimiter=’,’;
SASHELP.VMACRO: OVERVIEW

- Contains a list of all the macro variables that currently exist in your SAS session
- Includes all the macro variables that have been defined by you and by the SAS system
## SASHELP.VMACRO: CONTENTS

<table>
<thead>
<tr>
<th>SCOPE</th>
<th>NAME</th>
<th>OFFSET</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL</td>
<td>MAKE1</td>
<td>0</td>
<td>Acura</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE10</td>
<td>0</td>
<td>GMC</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE11</td>
<td>0</td>
<td>Honda</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE12</td>
<td>0</td>
<td>Hummer</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE13</td>
<td>0</td>
<td>Hyundai</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE14</td>
<td>0</td>
<td>Infiniti</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE15</td>
<td>0</td>
<td>Isuzu</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE16</td>
<td>0</td>
<td>Jaguar</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE17</td>
<td>0</td>
<td>Jeep</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE18</td>
<td>0</td>
<td>Kia</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE19</td>
<td>0</td>
<td>Land Rover</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE2</td>
<td>0</td>
<td>Audi</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE20</td>
<td>0</td>
<td>Lexus</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE21</td>
<td>0</td>
<td>Lincoln</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE22</td>
<td>0</td>
<td>MINI</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE23</td>
<td>0</td>
<td>Mazda</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE24</td>
<td>0</td>
<td>Mercedes-Benz</td>
</tr>
<tr>
<td>GLOBAL</td>
<td>MAKE25</td>
<td>0</td>
<td>Mercury</td>
</tr>
</tbody>
</table>
SASHELP.VMACRO: EXAMPLE 1 CODE

proc print data=sashelp.vmacro(where=(name='MAKE'));
run;
%macro delvars;

data vars;
   set sashelp.vmacro;
run;

data _null_;  
   set vars;
   temp=lag(name);
   if scope='GLOBAL' and substr(name,1,3) ne 'SYS' and temp ne name 
      then 
         rc=dosubl('%symdel '|'||trim(left(name))'||''); 
   run;
%mend;

%delvars
%let x=1;
%let y=2;
%let z=3;
%put &=x &=y &=z;
X=1 Y=2 Z=3

%delvars

NOTE: There were 71 observations read from the data set SASHELP.VMACRO.

NOTE: The data set WORK.VARS has 71 observations and 4 variables.

NOTE: DATA statement used
(Total process time):
  real time 0.04 seconds
  cpu time 0.04 seconds

NOTE: There were 71 observations read from the data set WORK.VARS.

NOTE: DATA statement used
(Total process time):
  real time 0.17 seconds
  cpu time 0.03 seconds

WARNING: Apparent symbolic reference X not resolved.
WARNING: Apparent symbolic reference Y not resolved.
WARNING: Apparent symbolic reference Z not resolved.

%put &=x &=y &=z;
CONTACT INFORMATION

Kevin Russell
SAS Institute Inc.
Email: support@sas.com
Web: support.sas.com
IT’S POP QUIZ TIME !!!
QUESTION 1

CALL SYMPUT is a new and improved version of CALL SYMPUTX.

True?

False?
QUESTION 2

Which style of comment is the recommend style comment to use within a macro definition?

A) An asterisk-style comment
B) A macro-style comment
C) A PL/1 style comment
QUESTION 3

Which system option will show you if your %IF condition was true or false?

A) SYMBOLGEN

B) MPRINT

C) MLOGIC
QUESTION 4

This is the correct syntax needed to use the macro IN operator?

options minoperator mindelimiter='‚';

%macro test(make);
  %if &make in(Acura BMW Infiniti Lexus %str(Mercedes-Benz))
    %then %let cartype=luxury;
    %let cartype=luxury;
  %put &=cartype;
%mend;

%test(BMW)

True?  False?

Bonus Question: What’s wrong with this syntax?
QUESTION 5

This syntax will create a read-only macro variable:

%let x=I love fishing! / readonly;

True?
False?

Bonus question:
How would I create a read-only macro variable?
QUESTION 6

Which system option do I have to set to *enable* the stored compiled macro facility?

A) MAUTOSOURCE
B) MFILE
C) MSTORED
D) MLOGIC
QUESTION 7

The following is the correct syntax needed to use the MFILE system option?

Filename myfile ‘c:\mycode\code.sas’;
Options mprint mfile;

%macro test;
    data new;
    set sashelp.cars;
    run;
%mend;
%test

True?  Bonus question: What’s wrong with this syntax?

False?
QUESTION 8

Which line of syntax is *incorrect*?

A) %put _user_;
B) %put &=x;
C) %put the value of the macro variable x is:  x;
D) They are all syntactically correct