Stump the Programmer #66

Counting Missing Values

Art Carpenter

In this Stump the Programmer problem, the author of this DATA step wants to count the number of missing values in a list of variables. The programmer has chosen to use the NMISS function, which he calls the "Number MISSing" function. Does he get the right answer? Why not? Can you suggest a better solution (or two)?

```
* What is the number of missing values?;
data a;
x=1;
y=.;
a='a';
b=' ';
misscnt=nmiss(x,y,a,b);
put misscnt=;
run;
```

Stump the Programmer #67

Bad Variable Name Choices

Art Carpenter

```
***** Part 1;
* Keyword DESCENDING - what is the
* order of the data after the SORT?;
data class;
   set sashelp.class;
   descending = age;
   run;

proc sort data=class;
   by descending name;
   run;
```

What happens if we inadvertently use a keyword as a variable name, and then use that variable in a PROC step that also uses that keyword? This problem was suggested by Howard Schreier on a SAS Forums thread.

The data set CLASS has a variable named DESCENDING. When that variable is used in a BY statement, how will the data be sorted?

```
***** Part 2;
* Keywords _NUMERIC_ _CHARACTER_ _ALL_;
* What variables are printed and why?;
proc summary data=sashelp.class nway ;
   class sex ;
   output out=badnames(rename = (_freq_ = _n_ _type_=_error_) )
      min(age) = _numeric_
      max(age) = _character_
      mode(age)= _all_       ;
      run ;

title BAD Names;
proc print data=badnames;
var _numeric_ _character_ _char_ _all_ ;
run ;
```

The variable short cuts _NUMERIC_, _CHARACTER_, and _ALL_, along with the temporary variable names _N_ and _ERROR_, are used as variable names in the data set BADNAMES. The variable shortcuts are valid keywords in a PROC PRINT's VAR statement. What variables are printed? Give yourself extra credit if you can determine the variable order in the output.

Stump the Programmer #68

By Art Carpenter

DO Syntax Has a Missing Semicolon

In the second step below the DO statement is missing a semicolon. What happens and why?

```
* Conditional DO block;
data notes;
   set sashelp.class;
   if sex='F' then do;
      note1=1;
      gender='Female';
      end;
   run;
proc print data=notes;
run;

* DO block has a missing semi-colon;
* What errors? Does anything change?;
data notes2;
   set sashelp.class;
   if sex='F' then do
      note1=1;
      gender='Female';
      end;
   run;
proc print data=notes2;
run;
```

Stump the Programmer #69

By Art Carpenter

Two Dates are Equal

Two date variables are created in a DATA step with equal values. These variables are then used to create to macro variables using a PROC SQL step. When they are compared using a macro %IF, they are found to be unequal. Why? What went wrong?

```
* two dates are equal;
* both are written to macro variables using SQL;
* The %IF finds them to be not equal - why??;

%macro test;
data a;
dt1 = date();
dt2 = date();
format dt1 date9.;
run;
proc sql noprint;
   select dt1, dt2
      into: date1, :date2
         from a;
quit;
%if &date1 = &date2 %then %put Dates are Equal;
%else %put Dates are not equal;
%mend test;
%test
```

The LOG shows:

```
Dates are not equal
```

Stump the Programmer #70

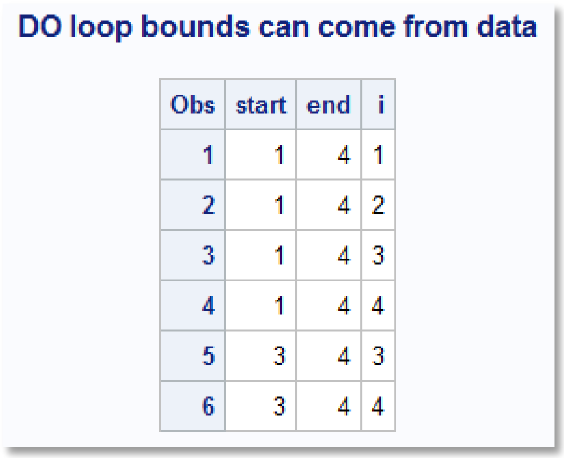
By Art Carpenter

Changing DO Loop Bounds

The upper and lower bounds of a DO loop can be set by the data.

```
data range;
start=1; end=4;output;
start=3; end=4;output;
run;

data test;
set range;
do i = start to end;
   output;
end;
run;
title1 'DO loop bounds can come from data';
proc print data=test;
run;
```



DO loop bounds can come from data

Obs	start	end	i
1	1	4	1
2	1	4	2
3	1	4	3
4	1	4	4
5	3	4	3
6	3	4	4

What would happen if these bounds (START and END) where changed within the DATA step?

```
data test2;
set range;
do i = start to end;
   start = 3;
   end = 8;
   output;
end;
run;

proc print data=test2;
run;
```