

# Macro Mania

So you're a macro whiz? Not good enough! To solve these problems, you'll also need imagination and cleverness, with a little genius (or is it madness?) thrown into the mix. These problems are difficult, so you'll get the most out of it if you work on them ahead of time.

Don't ask for solutions! They don't exist, at least not in printed or electronic form. The only way to obtain the answers is to attend the presentation!

The problems are copyrighted. To request permission to reprint problems, or to offer comments or questions about these problems, call me at (781) 862-4642, or write to:

Bob Virgile  
Robert Virgile Associates, Inc.  
44 Woburn Street  
Lexington, MA 02420-2222

[rvirgile@comcast.net](mailto:rvirgile@comcast.net)

A few technical notes:

Assume that any named data sets or variables actually exist, and that there are no typographical errors. Unless a problem explicitly states otherwise, assume there are no error or warning messages. This implies that all `%if` and `%do` statements appear within a macro definition.

All words mean what they appear to mean. For example, statement style macros are not permitted. The code which precedes each problem is not permitted to begin an unfinished comment statement. While the problems may be tricky, the wording is intended to be as clear as possible.

## The Problems

### What's the Record in this Joint?

1. In the code below, the final statement generates an error message: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required. Explain the source of the error.

```
%let v1=5;

%let v2=3;

%let v3=2;

%let v4=1;

%let v5=4;

%put %eval(5+&&&&&&&v&&&&v&&v&v1.....);
```

### When the Going Gets Tough

2. These statements generate the same error message as the first problem did. Why?

```
%let slogan = *** Lowest Prices in Town ***;

%if &slogan = *** Lowest Prices in Town ***

    %then %put The Tough Go Shopping!;
```

### Where Did You Go?

3. Describe the error in this code. You can assume that the names are few enough and short enough that they fit into a single macro variable.

```
proc sql;

    select count(distinct(name)) into : you

        from allnames;

    select distinct(name) into : all&you

        separated by ' ' from allnames;

quit;
```

## Am I Being Evaluated Here?

4. Describe the outcome when this code attempts to execute.

```
%let i=0;

%let u=What about you?;

%do %until (%length(%scan(&u, &i+1, %str( )))=0);

    %let i = &i + 1;

%end;
```

## A Rose by Any Other Name

This is really a set of three progressively more difficult problems. In each case, when the key statements execute, they generate Match #2 but do not generate Match #1. How could this happen?

5. The key statements for problem 5:

```
%if "&flower" = "rose" %then %put Match #1;

%let flower = &flower;

%if "&flower" = "rose" %then %put Match #2;
```

6. The key statements for problem 6:

```
%if "&flower" = "rose" %then %put Match #1;

%let leaf = &leaf;

%if "&flower" = "rose" %then %put Match #2;
```

7. The key statements for problem 7:

```
%if "&flower" = "rose" %then %put Match #1;

%if "&flower" = "rose" %then %put Match #2;
```

## It Had to Happen Sooner or Later

For this problem, you actually have to write a program!

8. Construct a macro `%test` so that these programs generate different results:

```
%test          data _null_;  
               call execute('%test');  
               run;
```

## Champagne, Anyone?

9. The macro at the end of this problem presents a simplified version of a bubble sort. It expects to run with the name of a macro variable as its one and only parameter. That macro variable should contain a series of four strings, and executing the macro replaces the macro variable with an ordered version of those strings.

For example, consider this combination of statements:

```
%let attitude = this is too clever;  
%bubbly (attitude)  
%put &attitude;
```

The `%put` statement would write the words in sorted order:

```
clever is this too
```

Your mission: illustrate that the macro is not foolproof. Specifically, create two versions of a macro variable, each holding the same four strings in different orders. When supplied to the macro below, there must be no error messages, yet the final orders generated by the macro must be different.

Here is the macro:

```
%macro bubbly (macvar);  
  %local i j changes dummy  
    word1 word2 word3 word4;  
  %do i=1 %to 4;  
    %let word&i = %scan(&&macvar,&i,%str( ));  
  %end;  
  %do %until (&changes=No);  
    %let changes=No;  
    %do i=1 %to 3;  
      %let j=%eval(&i+1);  
      %if &&word&j < &&word&i %then %do;  
        %let changes=Yes;  
        %let dummy = &&word&i;  
        %let word&i = &&word&j;  
        %let word&j = &dummy;  
      %end;  
    %end;  
  %end;  
  %let &macvar=&word1 &word2 &word3 &word4;  
%mend bubbly;
```