Level 1

1. **Validating Macro Parameters**
	1. Open the program shown below into the Editor window and submit it.

**%macro custtype(type);**

 **%let type=%upcase(&type);**

 **proc print data=orion.customer\_dim;**

 **var Customer\_Group Customer\_Name Customer\_Gender**

 **Customer\_Age;**

 **where upcase(Customer\_Group) contains "&type";**

 **title "&type Customers";**

 **run;**

**%mend custtype;**

**%custtype(Internet)**

* 1. Modify the macro to use %IF-%THEN/%ELSE statements to validate the parameter TYPE. The macro will submit the PROC PRINT step only if the TYPE parameter has a value of GOLD or INTERNET. If the TYPE value is not correct, the macro should write this message to the SAS log:

ERROR: Value of TYPE: *xxxxx* is not valid.

ERROR: Valid values are INTERNET or GOLD

The value of *xxxxx* is the TYPE parameter.

**NOTE:** Be sure to set the appropriate option to activate the IN operator.

* 1. Resubmit the macro definition and call the macro using both valid and invalid parameter values.
	2. Modify the macro to test whether the value of TYPE is null. If the value is null, the PROC PRINT step should not execute, and the macro should write this message to the SAS log:

ERROR: You must provide a value for TYPE

ERROR: Valid values are INTERNET or GOLD

The macro should first check for a null value in TYPE. If TYPE is not null, then the macro should make the value of TYPE uppercase and test for valid values of GOLD or INTERNET.

* 1. Resubmit the macro definition using a null value, a valid value for TYPE, and an invalid value for TYPE.

Level 2

1. **Conditionally Processing Complete Statements**
	1. Open the program shown below into the Editor window and submit it.

**%macro listing(custtype);**

 **proc print data=orion.customer noobs;**

 **run;**

**%mend listing;**

**%listing(2010)**

* 1. Modify the macro to test the parameter CUSTTYPE. If the value is null, insert the following statement into the PROC PRINT step:

**var Customer\_ID Customer\_Name Customer\_Type\_ID;**

**title "A Listing of All Customers";**

 If the value is not null, insert the following statements into the PROC PRINT step:

**where Customer\_Type\_ID =&custtype;**

**var Customer\_ID Customer\_Name;**

**title "A Listing of &custtype Customers";**

* 1. Resubmit the macro definition and call the macro using a null value and a valid value of CUSTTYPE.
	2. Alter the macro definition to validate the value of CUSTTYPE against a data-driven list.
		1. Use the SQL procedure to create a macro variable named IDLIST that contains the unique values from the variable **Customer\_Type\_ID** found in the **orion.customer\_type** data set.
		2. Insert the %IF logic to test the value of CUSTTYPE against the value of IDLIST. If the value of CUSTTYPE is not a valid customer ID, create a macro variable named FLAG and assign the value of 1. If the value of CUSTTYPE is a valid customer ID or null, assign the value of 0 to the macro variable named FLAG.

🖉 In order to negate the returned value when using the IN operator, the NOT operator must precede the syntax so that the opposite value is returned.

 %IF NOT(&*macvar* IN &*valuelist*) %THEN … ;

* + 1. Insert the %IF logic to test the value of FLAG. If the value of FLAG is 0, execute the code written in part **b**. If the value of FLAG is 1, the PROC PRINT should not execute, and the macro should write this message to the SAS log:

Partial SAS Log

ERROR: Value for CUSTTYPE is invalid.

 Valid values are 1010 1020 1030 1040 2010 2020 2030 3010

* 1. Resubmit the macro definition using a null value, a valid value for CUSTTYPE, and an invalid value for CUSTTYPE.

Level 3

1. **Validating Parameters and Setting Error Codes**
	1. Open the program shown below into the Editor window and submit it.

**%macro generatecode(bartype=vbar, dims=3d,**

 **var=Customer\_Age\_Group, color=pink,**

 **surface=s);**

 **proc gchart data=orion.customer\_dim;**

 **&bartype&dims &var;**

 **pattern color=&color value=&surface;**

 **run;**

 **quit;**

**%mend generatecode;**

**%generatecode()**

* 1. Modify the macro to validate the parameters based on the following conditions:

| **Macro Variable**  | **Possible Values** |
| --- | --- |
| BARTYPE | VBAR, HBAR |
| DIMS | 3D, null  |
| SURFACE | S, X1, X2, X3, X4, X5 |

* + 1. Create an additional macro variable named NUMERRORS that accumulates the number of parameters errors.
		2. Only execute the PROC GHART code if the value of NUMERRORS is zero.
	1. The macro should write the following messages to the SAS log when a parameter is invalid:

Partial SAS Log

693 **%generatecode(bartype=sbar, dims=1t ,surface=99)**

ERROR: Invalid Bar Type was supplied.

 Valid Values are VBAR or HBAR.

ERROR: Invalid Dimension Value.

 The value can be 3D or a null value.

ERROR: Invalid Surface Value.

 The value can be S, X1, X2, X3, X4, X5.

ERROR: Due to parameter errors SAS code will not execute.

 You have 3 errors.

* 1. Alter the macro to generate the following warning message in the SAS log when crosshatching (X1-X5) is used with a three-dimensional chart. The PROC GHART step should execute.

694 %generatecode(surface=x4)

WARNING: A DIMS value of 3D requires Solid surface.

 Surface parameter will be reset to S

MPRINT(GENERATECODE): proc gchart data=orion.customer\_dim;

MPRINT(GENERATECODE): VBAR3D Customer\_Age\_Group ;

MPRINT(GENERATECODE): pattern color=pink value=S;

MPRINT(GENERATECODE): run;

MPRINT(GENERATECODE): quit;

NOTE: There were 77 observations read from the data set ORION.CUSTOMER\_DIM.

Level 1

1. **Using Macro Loops and Indirect References**
	1. Open the program shown below into the Editor window.

**proc means data=orion.order\_fact sum mean maxdec=2;**

 **where Order\_Type = 1;**

 **var Total\_Retail\_Price CostPrice\_Per\_Unit;**

 **title "Summary Report for Order Type 1";**

**run;**

* 1. Define a macro program that generates a separate PROC MEANS step for each of the order types in the **orion.order\_fact** data set. The values of **Order\_Type** range from 1 to 3.
	2. Insert the following DATA step into the macro definition to create a series of macro variables named TYPE*x*, where *x* is the value of \_N\_. The value of each TYPE macro variable should be the value of **LABEL**.

**data \_null\_;**

 **set orion.lookup\_order\_type end=last;**

 **call symputx('type'||left(\_n\_), LABEL);**

 **if last then call symputx('endloop', \_n\_);**

**run;**

The **orion.lookup\_order\_type** data set contains the variable **LABEL**, which describes the different order types.

Partial Listing of **orion.lookup\_order\_type**

 START LABEL

 1 Retail Sale

 2 Catalog Sale

 3 Internet Sale

* 1. Modify the macro to use the following:
* the ENDLOOP macro variable as the stop value for the iterative DO loop
* an indirect reference to TYPE*x* in the TITLE statement
* an **s** concatenated to the end of the value of TYPE in the TITLE statement

Summary Report for Retail Sales

Level 2

1. **Generating Data-Dependent Steps**
	1. Open the program shown below into the Editor window. This program creates a summary data set named **customer\_freq** that summarizes the variable **Total\_Retail\_Price** by **Customer\_ID** and then sorts by descending **sum**. CALL SYMPUTX creates a series of macro variables named TOP1 through TOP*x*, where *x* is the
	value of the macro variable OBS.

**%macro tops(obs=3);**

 **proc means data=orion.order\_fact sum nway noprint;**

 **var Total\_Retail\_Price;**

 **class Customer\_ID;**

 **output out=customer\_freq sum=sum;**

 **run;**

 **proc sort data=customer\_freq;**

 **by descending sum;**

 **run;**

 **data \_null\_;**

 **set customer\_freq(obs=&obs);**

 **call symputx('top'||left(\_n\_), Customer\_ID);**

 **run;**

**%mend tops;**

**%tops()**

**%tops(obs=5)**

* 1. Modify the macro to print a listing of the top *x* customers from the **orion.customer\_dim** data set. Display the variables **Customer\_ID**, **Customer\_Name**, and **Customer\_Type**. Use a macro loop to dynamically generate the values for the WHERE statement based on the macro variables TOP1 through TOP*x*.

Partial SAS Log

MPRINT(TOPS): proc print data=orion.customer\_dim noobs;

MPRINT(TOPS): where Customer\_ID in ( 16 10 45);

MPRINT(TOPS): var Customer\_ID Customer\_Name Customer\_Type;

MPRINT(TOPS): title "Top 3 Customers";

MPRINT(TOPS): run;

MPRINT(TOPS): proc print data=orion.customer\_dim noobs;

MPRINT(TOPS): where Customer\_ID in ( 16 10 45 2806 195);

MPRINT(TOPS): var Customer\_ID Customer\_Name Customer\_Type;

MPRINT(TOPS): title "Top 5 Customers";

MPRINT(TOPS): run;

Level 3

1. **Generating Multiple Macro Calls**
	1. Open the program shown below into the Editor window. Submit the program and review the result.

**%macro memberlist(custtype);**

 **proc print data=Orion.Customer\_dim;**

 **var Customer\_Name Customer\_ID Customer\_Age\_Group;**

 **where Customer\_Type="&custtype";**

 **title "A List of &custtype";**

 **run;**

**%mend memberlist;**

**%macro listall;**

 **data \_null\_;**

 **set orion.customer\_type end=final;**

 **call symputx('type'||left(\_n\_), Customer\_Type);**

 **if final then call symputx('n',\_n\_);**

 **run;**

 **%put \_user\_;**

**%mend listall;**

**%listall**

* 1. Modify the LISTALL macro to call the MEMBERLIST macro. The result of the macro call should create a PROC PRINT step for each customer type. Use a macro loop and indirect references to generate the appropriate macro calls.

Level 1

1. **Understanding Symbol Tables**

Without submitting the programs, identify in which symbol table the macro variable DOG is located.

* Assume that each example is submitted in a new SAS session.
	1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**%let dog=Paisley;**

**%macro whereisit;**

 **%put My dog is &dog;**

**%mend whereisit;**

**%whereisit**

* 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**%macro whereisit;**

 **%let dog=Paisley;**

 **%put My dog is &dog;**

**%mend whereisit;**

**%whereisit**

* 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**%macro whereisit(dog);**

 **%put My dog is &dog;**

**%mend whereisit;**

**%whereisit(Paisley)**

* 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**%macro whereisit;**

 **data \_null\_;**

 **call symputx('dog','Paisley');**

 **run;**

 **%put My dog is &dog;**

**%mend whereisit;**

**%whereisit**

* 1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**%macro whereisit;**

 **%local dog;**

 **data \_null\_;**

 **call symputx('dog','Paisley');**

 **run;**

 **%put My dog is &dog;**

**%mend whereisit;**

**%whereisit**

Level 2

1. **Controlling Macro Variable Storage**
	1. Open the program shown below into the Editor window.

**%macro varscope;**

 **data \_null\_;**

 **set orion.customer\_type end=final;**

 **call symputx('localtype'||left(\_n\_), Customer\_Type);**

 **if final then call symputx('localn',\_n\_);**

 **run;**

 **%put \_user\_;**

**%mend varscope;**

**%varscope**

* 1. Modify the program so that all macro variables created in the DATA step are stored in the **local** symbol table.
	2. Modify the program by adding the following statement before the DATA step and removing the scope specification in the SYMPUTX routine.

**%local x;**

In which symbol table are the macro variables stored? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Modify the program so that all macro variables created in the DATA step are stored in the **global** symbol table

.

Level 3

1. **Creating Multiple Symbol Tables**
	1. Open the **cleanup** program and submit the macro to delete all global macro variables.
	2. Open the program shown below into the Editor window.

**%macro createmacvar;**

 **data \_null\_;**

 **set orion.lookup\_order\_type end=last;**

 **call symputx('type'||left(start), LABEL,'L');**

 **if last then call symputx('endloop', \_n\_,'L');**

 **run;**

**%mend createmacvar;**

**%macro sumreport;**

 **%createmacvar**

 **%local num;**

 **%do num=1 %to &endloop;**

 **proc means data=orion.order\_fact sum mean maxdec=2;**

 **where Order\_Type = &num;**

 **var Total\_Retail\_Price CostPrice\_Per\_Unit;**

 **title "Summary Report for &&type&num";**

 **run;**

 **%end;**

**%mend sumreport;**

**%sumreport**

Submit the program. Review and describe the results.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Correct the program so that the SUMREPORT macro executes correctly and does not create any global macro variables. Verify that the title is resolving properly. In addition, add an **s** to the end of the type description in the title.

PROC MEANS Output

 Summary Report for Retail Sales

 The MEANS Procedure

Variable Label Sum Mean

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Total\_Retail\_Price Total Retail Price for This Product 44654.56 137.82

CostPrice\_Per\_Unit Cost Price Per Unit 11730.73 36.21

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Summary Report for Retail Sales

 The MEANS Procedure

Variable Label Sum Mean

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Total\_Retail\_Price Total Retail Price for This Product 44654.56 137.82

CostPrice\_Per\_Unit Cost Price Per Unit 11730.73 36.21

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

 Summary Report for Internet Sales

 The MEANS Procedure

Variable Label Sum Mean

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Total\_Retail\_Price Total Retail Price for This Product 21491.55 174.73

CostPrice\_Per\_Unit Cost Price Per Unit 5356.95 43.55

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ