**Question 1**

**The Macro Facility.**

The [SAS Macro Facility](https://documentation.sas.com/?docsetId=mcrolref&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en) is a code-generation system which uses variables and macros.

A macro is a program segment where variables are used instead of the names of data sets, data subject variables or numeric, text or date literals (constants) in the program. A macro processor processes the macro by substituting a variable with the text to which it refers. In effect a macro is a code template which results in a valid SAS program, after procession is complete.

When macros are declared, the '%' sign is used to distinguish the macro language from identically named SAS program keywords. When it is used, the macro's name is preceeded by the '%' sign.

When variables are declared, a name is used on its own. When the variable is to be referenced in code, the '&' symbol is used.

**Self-assessment question**

Consider the following code:

%let var= week;  
 %let index= **1**;  
   
 %let title= %str(This is a test!);  
 title "**&title**";  
 title2 "**&var.&index** data"**;**  
   
 **data** TEST;  
 input  
 week1  
 weight1  
 week2  
 weight2  
 ;  
 datalines;  
 15 70 25 74   
 ;  
 **run;**  
   
 **proc print** data= TEST noobs;  
 var **&var.&index;**  
 **run;**

Pretend to be the 'macro processor' and process the code by substituting variable names with their associated text. Paste the code you expect should run in the box below.

How might you determine whether you have processed the code in the same way as the macro processor?

*Hint*:  
Add: **options symbolgen;** before your code.  
Add: **options nosymbolgen;** after it.  
And then read the log carefully after running the program.

## ANSWER 1

/\*symbolgen resolves the macro variables in the sas log window\*/

options symbolgen;

/\*assign a macro variable to week and and weight and index for each\*/

%let var= week;

%let var\_weight=weight;

%let index= 1;

%let index2 =2;

/\* create a title for each output using macro\*/

%let title= %str(This is a test!);

title "&title";

title2 "&var.&index data";

**data** TEST;

/\* replace the previous data step with macro variables\*/

input

&var.&index.

&var\_weight.&index.

&var.&index2.

&var\_weight.&index2.

;

datalines;

15 70 25 74

;

**run**;

**%macro** ***print***;

proc print data= TEST noobs;

var &var.&index.

/\*confirm it works\*/

&var\_weight.&index.

&var.&index2.

&var\_weight.&index2;

run;

**%mend** print;

%***print***;

/\*specifies the log messages will not be displayed\*/

options nosymbolgen;

### Question 2

##### Macros

A macro begins with the statement: **%macro name(<variable-list>);**  
and ends with the statement: **%mend name;**  
where name is the name of the macro and <variable-list> is a comma separated list of variables.

A macro is 'called' by writing: **%name(l)**  
where **name** is the name of the macro and **l** is a comma-separated list of text elements to use to replace the corresponding macro variables.

One strategy for developing a macro is to start with correct code and then to use the features of the macro language to achieve the same result. This might mean replacing variables or literals with macro variables or it might mean using a macro statement.

###### Self-assessment question

Write and test a macro that produces the following code:

**data \_NULL\_**;  
 put '1';  
 put '2';  
 put '3';  
 put '4';  
 put '5';  
 **run;**

*Hints*:

* You may find it useful to read about the **%do ... %to ...** **%by** macro statement.
* When writing your code, you may find it useful to surround references to macro variables with quotes (**"**) to ensure the variable's text is expanded before being printed.
* You may find it useful to add:  
  **options symbolgen mprint mlogic;** before your 'call' to the macro.  
  **options nosymbolgen nomprint nomlogic;** after it.  
  And then to read the log carefully!

## ANSWER 2

/\* mprint returns the text generated by the output

while mlogic traces the flow of the exection in this case put as '1','2','3','4','5'\*/

options symbolgen mprint mlogic;

/\*create a macro variable create\*/

**%macro** create(put);

/\*use do loop to return the values of i from the put and by to define the sequence\*/

%do i=**1** %to &put %by **1** ;

data \_NULL\_;

%put &i;

run;

%end;

**%mend** create;

%***create***(**5**)

options nosymbolgen nomprint nomlogic;

### Question 3

##### Show the contents of a data set

The following code provides useful information about a SAS dataset:

ods select position**;**  
 **proc contents**   
 data=LOAN\_RISK  
 varnum  
 **;**  
 **run;**  
 ods select all**;**

**proc print** data=LOAN\_RISK(obs=**10**);  
 format  
 account  
 history  
 purpose  
 amount   
 savings  
 employment  
 married   
 debtors   
 resident  
 property  
 plans   
 housing   
 job   
 telephone  
 foreign   
 when\_assessed  
 rating  
 **;**  
 **run;**

**proc print** data=LOAN\_RISK(obs= 10) label**;**  
 **run;**

It uses **proc contents** to list the details of all the variables in the data set. Next it uses two **proc print** statements to display first the raw data and then the formatted data for the first 10 data subjects.

###### Self-assessment question

Can you you turn the above code into a macro that works for any data set?

*Hint*:  
The following macro creates macro variable called **ds\_vars**. (Do not worry too much about how it works but if you want an explanation, it can be found in the [SAS documentation](https://documentation.sas.com/?docsetId=mcrolref&docsetTarget=n0na9f3e0qxdazn1lh6o7s919lrq.htm&docsetVersion=9.4&locale=en).) **ds\_vars** contains a list of all the variables in a data set.

To process the data set **TEST** created in question 1, you call the macro using the statement:  
**%lst(TEST);**   
after which, **ds\_vars** would contain the following list:  
week1 weight1 week2 weight2

%macro lst(dsn);  
 %local dsid cnt rc;  
 %global ds\_vars;  
 %let ds\_vars=;  
 %let dsid= %sysfunc(open(**&dsn**));  
 %if **&dsid** ne **0** %then %do;   
 %let cnt= %sysfunc(attrn(**&dsid**, nvars));  
 %do i= **1** %to **&cnt**;  
 %let ds\_vars=**&ds\_vars** %sysfunc(varname(**&dsid**,**&i**));  
 %end;  
 %put ds\_vars= **&ds\_vars**;  
 %end;  
 %else %put **&dsn** cannot be opened.;  
 %let rc= %sysfunc(close(**&dsid**));  
 %mend lst;

## ANSWER 3

options symbolgen mprint mlogic;

%let dsn = LOAN\_RISK;

/\* process the data using ds\_vars\*/

**%macro** lst(dsn);

%local dsid cnt rc;

%global ds\_vars;

%let ds\_vars=; /\*call all variables\*/

%let dsid= %sysfunc(open(&dsn));

%if &dsid ne **0** %then %do;

%let cnt= %sysfunc(attrn(&dsid, nvars));

%do i= **1** %to &cnt;

%let ds\_vars=&ds\_vars %sysfunc(varname(&dsid,&i));

%end;

%put ds\_vars=ds\_vars;

%end;

%else %put &dsn cannot be opened.;

%let rc= %sysfunc(close(&dsid));

**%mend** lst;

%***lst***(LOAN\_RISK)

%put macro variable ds\_vars =&ds\_vars;

/\* create a macro variable that outputs many procedures\*/

**%macro** ***many***;

ods select position;

proc contents data=&dsn

varnum;

title "DATA SET &dsn";

run;

ods select all;

proc print data=&dsn (OBS=**10**);

format &ds\_vars;

run;

proc print data =&dsn (OBS=**10**)label;

run;

**%mend** many;

%***many***;

options nosymbolgen nomprint nomlogic;

### Question 4

##### List extreme observations

The following code lists all the data for a data subject if the subject falls into the set of those with extreme values for a particular variable:

ods csvall file= '/folders/myfolders/imat5168/2020/age\_extremeobs.csv';  
 ods select extremeobs**;**  
 **proc univariate** data= LOAN\_RISK nextrobs= **10**;  
 var age;  
 id customer**;**  
 **run;**  
 ods select all;  
 ods csvall close**;**

**data** AGE\_EXTREME;  
 infile  
 '/folders/myfolders/imat5168/2020/age\_extremeobs.csv'  
 firstobs= **6**  
 dsd  
 ;  
 input  
 value  
 customer  
 obs  
 @@  
 **;**  
 **run;**

**proc sql;**  
 create table AGE\_EXTREMES as  
 select \*   
 from LOAN\_RISK  
 where customer in (  
 select customer  
 from AGE\_EXTREME  
 )  
 order by age desc**;**  
 **quit;**  
   
 **proc print** data= AGE\_EXTREMES label noobs**;**  
 **run;**

###### Self-assessment question

Can you turn the above code into a macro that works for all data sets?

*Hint*:  
You may find the following macro declaration useful:

%macro show\_extreme\_obs(  
 data= */\* data set \*/*  
 , var= */\* variable \*/*  
 , nobs= **5** */\* number of observations \*/*  
 , id= */\* primary key \*/*  
 , idtype= */\* id type - use $ to signify a text key \*/*  
 );

This declaration shows an alternative method for declaring macro variables. Without the '=' sign, macro variables are positional; the first variable in the list maps onto the first text element in the list when the macro is 'called'.

With the '=', macro variables are used only when their name and value are provided in the 'call'. This form also means that default values can be used to avoid the user having to specify a value if the default is satisfactory; the declaration of nobs demonstrates this.

The macro declaration can be called as follows:  
**%show\_extreme\_obs(data= NPDA, var= target1, id= unit, idtype= $);**  
or:  
**%show\_extreme\_obs(data= NPDA, var= target1, nobs= 10, id= unit, idtype= $);**

## ANSWER 4

options symbolgen mprint mlogic;

/\*create a macro variable that reads work.LOAN\_RISK, Create a csv file for age extreme obs\*/

**%macro** createtxtfl(dsn,age,customer);

ods csvall file= 'C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104\Documents\IMAT5168 Analytical Programming Mark\Lab 10\age\_extremeobs.csv';

ods select extremeobs;

proc univariate data= &dsn nextrobs= **10**;

var &age;

id &customer;

run;

ods select all;

ods csvall close;

**%mend** createtxtfl;

%***createtxtfl***(dsn=work.LOAN\_RISK,age=age,customer=customer);

/\*create a data step to import Age Extreme Obs\*/

%let path=C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104\Documents\IMAT5168 Analytical Programming Mark\Lab 10\;

%let var = Value;

%let var2 = customer;

%let var3 = Obs;

/\*create a macro variable for data, var delimiter\*/

**%macro** show\_extreme\_obs(data,flname,delimiter);

data &data;

infile

"&path&flname..csv" /\*call the macro variables path and flname to get age extreme obs data set\*/

dlm=&delimiter

firstobs=**8**

;

input

var

var2

var3

@@

;

run;

**%mend**;

%***show\_extreme\_obs***(AGE\_EXTREME,age\_extremeobs,",");

/\*create a macro varibale using into statement in SQL\*/

**proc** **sql**;

create table AGE\_EXTREMES as

select \*

into :name separated by " "

from LOAN\_RISK

where customer in (

select customer

from AGE\_EXTREME

)

order by age desc;

**quit**;

**%macro** printed (data);

proc print data= &data label noobs;

run;

**%mend**;

%***printed***(AGE\_EXTREMES);

options nosymbolgen nomprint nomlogic;

### Question 5

##### Test Residuals for Normality

The one-way ANOVA analysis provides an output statement, which can be used to provide data to help assess the validity of the statistical test as follows:  
output out= RESIDUALS r= residual p= predicted;  
This statement creates a dataset called RESIDUALS, containing two variables: residual and predicted.

###### Self-assessment question

Write a macro to perform a one-way ANOVA on any data set, which includes not only the statistical test but also provides an assessment of whether the residuals follow a normal distribution. You should be able to specify the name of the data set, the classifier variable and the calculated variable.

## ANSWER 5

options symbolgen mprint mlogic;

/\*create %let global options\*/

%let title = Ames 1978;

%let title2 = One-way ANOVA Analyis of;

%let ventvar =ventilation;

%let fol = folate;

%let dsn = amess1978;

%let out =RESIDUALS;

%let pred = predicted;

%let redl = residual;

title "&title2 &title showing &ventvar and &fol";

**data** &dsn;

input

&ventvar

&fol

@@

;

datalines;

1 243 2 206 3 241

1 251 2 210 3 258

1 275 2 226 3 270

1 291 2 249 3 293

1 347 2 255 3 328

1 354 2 273

1 380 2 285

1 392 2 295

2 309

;

**run**;

**proc** **means**

data = &dsn n mean std fw=**5** maxdec=**1** nonobs;

class &ventvar;

var &fol;

**run**;

**proc** **univariate**

data = &dsn;

class &ventvar;

var &fol;

qqplot /normal(mu=est sigma=est);

histogram /normal;

**run**;

**proc** **sgplot**

data = &dsn;

\* scatter x=ventilation y=folate;

vbox &fol / group=&ventvar;

xaxis min=**0** max=**4**;

yaxis min=**0**;

**run**;

/\*observe the calculated variable folate by class variable ventilation\*/

**proc** **anova**

data = &dsn;

class &ventvar;

model &fol=&ventvar;

means &ventvar /lsd;

**run**;

/\* proc glm to get the least squares to fit general linear models\*/

**proc** **glm**

data = &dsn;

class &ventvar;

model &fol=&ventvar;

means &ventvar /lsd hovtest=bartlett;

output out=&out p=&pred r=&redl;

**run**;

/\*proc univariate to see if resdiuals follow a normal distribution\*/

**proc** **univariate**

data=&out

plot normal;

var &redl;

qqplot /normal(mu=est sigma=est);

histogram /normal;

**run**;

/\* specified a scatter plot to view the correlation\*/

**proc** **sgplot**

data=&out;

scatter x=&pred y=&redl;

refline **0** / axis=y;

**run**;

options nosymbolgen nomprint nomlogic;

### Question 6

##### Portability

Throughout this module, the assumption has been that SAS datasets will be stored in the default WORK library. This means that when the SAS interface is closed, the dataset will be deleted. You may not always want this to be the case, especially if it takes a long time to run the code to create the dataset. It would be useful to be able to specify an alternate directory.

Similarly, the location of raw data files and the location of your code libraries may vary by the SAS system you are using and by the project you are working on.

One way of handling this problem is to use a driver program to define the macro variables containing information about the locations of important files for a project. When this is run, it provides the macro variables for all subsequent code to use. This will mean re-writing your programs only to use references to these macro variables.

As with all macros, it is convenient to create and debug the program using the WORK library and temporary local folders for the raw data and the code. When everything works as expected, the explicit locations can be replaced by appropriate macro variables and the program run from within the driver program.

###### Self-assessment question

Review slides 9-12 in the summary slide set on portability to see how a driver program might be constructed.

Re-write your program for importing and merging the credit data sets, so that it can read the raw data from any directory, can include program files (for example containing macros) from any library directory and can put the resulting SAS data sets in any appropriate directory.

*Hint*: You will know you have succeeded, if is it possible to move all the code, raw data and data sets to a new location, only by changing macro variables.

ANSWER 6 (FAILED ATTEMPT)

FILE DIRECTORY FOR THE OUTPUT WAS CREATED BUT THE OUTPUT STILL SAVES TO WORK LIBRARY

options symbolgen mprint mlogic;

/\* The included file should use &library to refer to the library where SAS files are to be stored \*/

%let library = MacroT;

libname &library "&data\_dir";

options fmtsearch=(&library);

/\* The driver file should use libs(file.sas) to refer to the library where SAS code files are stored \*/

/\*%let code\_dir = %str(C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104\Documents\IMAT5168 Analytical Programming Mark\Lab 7);

filename libs "&code\_dir";

%include libs(Question 6.sas);

/\*

psa= 06

requirement= create LOAN-RISK data set

requirement= merge customer with ratings and assessment dates

changed= 2020.03.01

\*/

/\*

requirement= set locale to UK

\*/

options locale = English\_UnitedKingdom;

/\*

requirement= set current working directory

\*/

/\*filename cwd 'C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104 (vfiler1.lec-admin.dmu.ac.ukHome4)\Documents\IMAT5168 Analytical Programming Mark\Lab 6'; \*/

%let data\_dir = %str(C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104\Documents\IMAT5168 Analytical Programming Mark\Lab 7);

%let data\_file = %str(psa06-credit.dat);

%let data\_file2 = %str(psa06-credrating.dat);

%let data\_file3 = %str(psa06-creddates.csv);

filename data "&data\_dir";

%let sas\_data = customer;

%let sas\_data2 = rating;

%let sas\_data3 = assessment;

/\*

requirement= import credit.dat into CUSTOMER

note= indentation strategy: new material on a new line

\*/

**data** &sas\_data; /\* capitalization makes data set name stand out \*/

infile

data(&data\_file)

missover

;

input /\* comments unnecessary due to labels and formats \*/

customer **1**-**3**

account **5**

duration **7**-**8**

history **10**

purpose $ **12**-**13**

amount **14**-**18**

savings **20**

employment **22**

instalment **24**

married **26**

debtors **28**

resident **30**

property **32**

age **34**-**36**

plans **38**

housing **40**

credits **42**

job **44**

dependents **46**

telephone **48**

foreign **50**

;

**run**;

/\*

requirement= import credrating.dat into RATING

\*/

/\*

requirement= create informat for rating

\*/

**proc** **format**;

invalue rating

'bad' = **1**

'good' = **0**

other = **.**

;

**run**;

**data** &sas\_data2;

infile

data(&data\_file2)

;

informat

rating rating.

;

input

customer

rating

;

**run**;

**data** &sas\_data3;

infile

data(&data\_file3)

delimiter = ','

firstobs = **2**

;

input

customer

day

month

year

;

when\_assessed = mdy(month, day, year);

drop

day

month

year

;

**run**;

/\*

requirement= join CUSTOMER, ASSESSMENT, RATING using customer id

requirement= set null values

requirement= apply labels

requirement= apply outformats

note= creates: LOAN\_RISK

\*/

/\*

requirement= sort CUSTOMER by customer id

note: overwrites CUSTOMER

\*/

**proc** **sort** data = &sas\_data out = CUSTOMER;

by customer;

**run**;

/\*

requirement= sort CUSTOMER by customer id

note: overwrites RATING

\*/

**proc** **sort** data = &sas\_data2 out = RATING;

by customer;

**run**;

/\*

requirement= sort ASSESSMENT by customer id

note: overwrites ASSESSMENT

\*/

**proc** **sort** data = &sas\_data3 out = ASSESSMENT;

by customer;

**run**;

/\*

requirement= format variables

note= data set: LOAN\_RISK

\*/

**proc** **format**;

value cr\_account

**1** = '< Â£0'

**2** = 'Â£0 - Â£200'

**3** = '>= Â£200'

**4** = 'no account'

**.** = 'missing'

other = 'ERROR'

;

value cr\_history

**0** = 'all paid'

**1** = 'bank paid'

**2** = 'paid before'

**3** = 'delay in paying'

**4** = 'not paid'

**.** = 'missing'

other = 'ERROR'

;

value $cr\_purpose

'0' = 'car (new)'

'1' = 'car (used)'

'2' = 'furniture etc'

'3' = 'radio/tv'

'4' = 'appliances'

'5' = 'repairs'

'6' = 'education'

'7' = 'vacation'

'8' = 'retraining'

'9' = 'business'

'X' = 'others'

'' = 'missing'

other = 'ERROR'

;

value cr\_savings

**1** = '<Â£100'

**2** = 'Â£100 - Â£500'

**3** = 'Â£500 - Â£1000'

**4** = '>= Â£1000'

**5** = 'no account'

**.** = 'missing'

other = 'ERROR'

;

value cr\_employment

**1** = 'unemployed'

**2** = '< 1 year'

**3** = '1 - 4 years'

**4** = '4 - 7 years'

**5** = '>= 7 years'

**.** = 'missing'

other = 'ERROR'

;

value cr\_married

**1** = 'male :was married'

**2** = 'female :is or was married'

**3** = 'male :single'

**4** = 'male :is married'

**5** = 'female :single'

**.** = 'missing'

other = 'ERROR'

;

value cr\_debtors

**1** = 'none'

**2** = 'co-applicant'

**3** = 'guarantor'

**.** = 'missing'

other = 'ERROR'

;

value cr\_resident

**1** = '1 year'

**2** = '2 years'

**3** = '3 years'

**4** = '>=4 years'

**.** = 'missing'

other = 'ERROR'

;

value cr\_property

**1** = 'real estate'

**2** = 'if not 1: building society loan'

**3** = 'if not 1/2: car or other'

**4** = 'no property'

**.** = 'missing'

other = 'ERROR'

;

value cr\_plans

**1** = 'bank'

**2** = 'stores'

**3** = 'none'

**.** = 'missing'

other = 'ERROR'

;

value cr\_housing

**1** = 'rent'

**2** = 'own'

**3** = 'for free'

**.** = 'missing'

other = 'ERROR'

;

value cr\_job

**1** = 'unemployed'

**2** = 'unskilled'

**3** = 'skilled employee'

**4** = 'management'

**.** = 'missing'

other = 'ERROR'

;

value cr\_telephone

**1** = 'yes'

**2** = 'no'

**.** = 'missing'

other = 'ERROR'

;

value cr\_foreign

**1** = 'yes'

**.** = 'no'

other = 'ERROR'

;

value cr\_rating

**1** = 'bad'

**0** = 'good'

**.** = 'missing'

other = 'ERROR'

;

**run**;

%let new\_data = LOAN\_RISK;

**data** &new\_data;

merge CUSTOMER RATING ASSESSMENT;

by customer;

if history = **9** then history = **.**;

if age = **999** then age = **.**;

if married = **9** then married = **.**;

if debtors = **9** then debtors = **.**;

if resident = **9** then resident = **.**;

if plans = **9** then plans = **.**;

if housing = **9** then housing = **.**;

if job = **9** then job = **.**;

if telephone = **9** then telephone = **.**;

label

customer = 'Customer ID'

account = 'Chequing account [overdraft?]'

duration = 'Duration in months'

history = 'Credit history'

purpose = 'Purpose [of loan?]'

amount = 'Credit amount [requested?]'

savings = 'Savings accounts/bonds'

employment = 'Present employment since'

instalment = 'Instalment rate % income'

married = 'Personal status and sex'

debtors = 'Other debtors/guarantors'

resident = 'Present residence since'

property = 'Property [purchase method?]'

age = 'Age in years'

plans = 'Other instalment plans'

housing = 'Housing [ownership?]'

credits = 'Number of existing credits'

job = 'Job [type?]'

dependents = 'Number of dependents'

telephone = 'Telephone [line rental?]'

foreign = 'Foreign worker'

when\_assessed = 'Date of assessment'

rating = 'Credit rating'

;

format

account cr\_account.

history cr\_history.

purpose $cr\_purpose.

amount nlmnlgbp8.0

savings cr\_savings.

employment cr\_employment.

married cr\_married.

debtors cr\_debtors.

resident cr\_resident.

property cr\_property.

plans cr\_plans.

housing cr\_housing.

job cr\_job.

telephone cr\_telephone.

foreign cr\_foreign.

when\_assessed nldate10.

rating cr\_rating.

;

**run**;

/\*

requirement= feedback about variable: type, size

note= exclude: enginehost

note= order: variable creation

\*/

ods exclude enginehost;

**proc** **contents**

data=LOAN\_RISK

varnum

;

**run**;

ods select all;

/\*

requirement= feedback about variable: value

note= observations limit: 10

note= format: removed temporarily

\*/

**proc** **print** data=&new\_data(obs=**10**);

format

account

history

purpose

amount

savings

employment

married

debtors

resident

property

plans

housing

job

telephone

foreign

when\_assessed

rating

;

**run**;

/\*

requirement= print records for first 10 customers

note= label: included

note= format: included

\*/

**proc** **print** data=&new\_data label noobs;

**run**;

options nosymbolgen nomprint nomlogic;