ASSESSMENT 4

QUESTION 1

List format data

When importing CREDIT.DAT, the specification stated that the data were in **column-format**, where each variable lay between fixed locations in the input buffer and all data were on a single line. (Here, a *line* is taken to mean the raw data between:

- the start of a file and the next line-terminator
- two successive line-terminators
- a line-terminator and the end of the file

where a *line-terminator* is one or more characters, used to mark the end a line of text in that particular file.)

In contrast to column-format data, **list-format** data is formed of a sequence where the raw data for successive variables is separated by one or more characters. Each variable's raw data may also be surrounded by quotes (").

To import list-format data, at the least you should specify the list of variables, using the *input* statement.

Self-assessment question

Open the CREDIT.DAT data set in Notepad++ and select: "View > Show Symbol > Show All Characters" from the menu. Also open the data set specification.

1. Determine whether a single character separates each of the variables. If there you can identify one character, go on to the next step.
2. Make a copy of the program you created to read all the data in column-format and alter the data step to import all the data in CREDIT.DAT, using list-format.
3. Either paste the updates to your program in the box below or write: "No single character separates the variables."

Answer 1

1. Yes- a single character (space) separates each variable. On the Show All Characters window, the variables are separated by red dots.
2.

```
options locale = English_UnitedKingdom;
```

```
filename pwd 'C:\Users\Folashikemi\OneDrive - De Montfort
University\P2586104 (vfiler1.lec-admin.dmu.ac.ukHome4)\Documents\IMAT5168
Analytical Programming Mark\Lab 2';

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'
        ;
run;
```

```
data CUSTOMER;
infile
            pwd(credit.dat)
            missover
      ;
      /* using list format to input variables
      note: list-format returns variables in the order they are inputed*/
      input
            customer
            account
            duration
            history
            purpose $
            amount
            savings
            employment
            instalment
            married
            debtors
            resident
            property
            age
            plans
            housing
            credits
            job
            dependents
            telephone
            foreign
      ;
      label
            customer    = 'ID Number'
            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'
      ;
      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.
			;
run;

ods exclude enginehost;
proc contents
		data=CUSTOMER
		varnum
		;
run;
ods select all;

proc print data=CUSTOMER(obs=10);
		format
				account
				history
				purpose
				amount
				savings
				employment
				married
				debtors
				resident
				property
				plans
				housing
				job
				telephone
				foreign
		;
run;

/*
		requirement= feedback about variable: value
		note= observations limit: 10
		note= label: present
		note= observation number: removed
*/

proc print
		data=CUSTOMER(obs=10)
		label
		noobs
		;
run;
```

QUESTION 2

Self-assessment question

Monthly exchange rates used by the UK HMRC (HMRC Exchange Rates (02/2020)) are presented in a CSV file: EXRATES-MONTHLY-0220.CSV in this content folder. Using ***only*** the data statement, import the CSV file into a SAS data set called: MEXRATE202002.
When your program works correctly, paste it into the box below.

Hints: It is suggested that you read up about:

1. the *infile* statement's DSD and FIRSTOBS options
2. the data step's LENGTH statement

ANSWER 2

```
filename pwd 'C:\Users\Folashikemi\OneDrive - De Montfort 
University\P2586104 (vfiler1.lec-admin.dmu.ac.ukHome4)\Documents\IMAT5168 
Analytical Programming Mark\Lab 4';
data MEXRATE202002;
/*firstobs=2 to start reading the data from the second observation*/
/* using dsd (delimiter-sensitive data) statement to separate the 
variable*/
infile pwd (exrates-monthly-0220.CSV) firstobs=2 dsd;
/*use the length statement is input values with longer than 8 characters 
I chose 50 just as an assumption incase i miss a value*/
length
      country$ 50.
      currency$ 50.
;
input
      Country $
      Currency $
      Currency_Code $
      Currency_Unit_per_pound
      Start_Date
      End_Date
;
/* use informat to convert non-standard data to standard data */
informat
      Start_Date   DDMMYY10.
      End_Date     DDMMYY10.
;
/*use format to convert standard data to nonstandard data*/
format
      Start_Date   DDMMYY10.
      End_Date     DDMMYY10.
;
run;
proc print
      data = MEXRATE202002;
      var Country
      Currency
      Currency_Code
      Currency_Unit_per_pound
      Start_Date
      End_Date
;
run;
```

QUESTION 3
proc IMPORT

The import procedure can make importing data in standard list formats easier.
Self-assessment question

After reviewing the documentation compare using the proc IMPORT procedure to import: EXRATES-MONTHLY-0220.CSV with the code you wrote previously.
Also, when your program works correctly, paste it into the box below.

ANSWER 3
```
/*using a macro variable named file*/
%let file=C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104
(vfiler1.lec-admin.dmu.ac.ukHome4)\Documents\IMAT5168 Analytical
Programming Mark\Lab 4;
options validvarname=v7;
proc import
datafile="&file\exrates-monthly-0220.CSV"
/*use rename statement to correct variable with VAR4 as column name*/
      out=MEXRATE202002 (rename=(VAR4 = Currency_Unit))
      dbms=csv
      /* to replace exsisting files */
      replace
;
/*to prevent sas from shorting the values*/
guessingrows=max;
run;
proc print data = MEXRATE202002
label
noobs
;
run;
```

QUESTION 4
Spreadsheets

The National Hip Fracture Database collects data about UK health services in its "Facilities audit." The 2019 Annual Report of Facilities Data is published as a spreadsheet, for which data definitions can also be obtained - although these are not necessary for this question.
Self assessment question

Using proc IMPORT, import the NHFD2019.XLXS spreadsheet. Is the result convenient to use when analysing data?

Don't forget to paste your working code into the box below.

Hint: Consider the length of variable names and whether all variables have names.

ANSWER 4

1. Initially, I tired using the proc import method I saw form SAS documentation.
   The column names were truncated, and I tired modifying the variable names using the Data Step.

The process was cumbersome and I though of how impossible it will be to adopt the same process for a larger data set.
However, upon further research, I learnt how to create a macro variable and named it file.
Then I did a proc import for the xlsx file.
The variable names in output weren't truncated and my data reads correctly.
Overall, I find this more convenient that the infile statement because I didn't have to use the input statement for each variable.

```
/*using a macro variable named file*/
%let file=C:\Users\Folashikemi\OneDrive - De Montfort University\P2586104
(vfiler1.lec-admin.dmu.ac.ukHome4)\Documents\IMAT5168 Analytical
Programming Mark\Lab 4;
options validvarname=v7;
proc import
datafile="&file\NHFD2019.xlsx"
      out=NHFD2019
      dbms=xlsx
      /* to replace exsisting files */
      replace
;
proc print data = NHFD2019
label
noobs
;
run;
```

QUESTION 5

Multiple observations per input buffer line.

Amess, Burman and Rees (1978) performed an experiment to measure the effect of nitrous oxide on bone-marrow function. For a part of the experiment evaluating how different ventilation strategies affected folate levels, the data are:

```
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
           ;
```

There are up to 3 subjects per data line. For each subject, the observations are:

1. Type of ventilation: numeric value.
2. Folate level: numeric value

Self assessment question

1. Explain how to import these data.
2. Import the data and paste your code in the box below, when you have been successful.
Hint: Review the input statement's '@@' option.

ANSWER 5

1. Suppose we have more than one observation on a single line of data, @@ will be included after the input statement to help SAS read multiple data on a single line.

```
data Experiment1978;
      input
            ventilation
            folate
            @@ /* more data on the same line */
      ;
      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 print
      data=Experiment1978
      label
      noobs
      ;
run;
```

QUESTION 6

Hierarchical data

Hierarchical data means that an indicator variable determines which data are read using one or more input statements.

Self assessment question

Create a suitable input statement for these modified physical fitness data lines. The first variable is new and is the group of the participant. All the other variables are the same as before. You should read only the data for the 'g' group.

```
      datalines;
r 44 89.47 44.609 11.37 62 178 182    g 40 75.07 45.313 10.07 62 185 185
g 44 85.84 54.297  8.65 45 156 168    b 42 68.15 59.571  8.17 40 166 172
r 38 89.02 49.874  9.22 55 178 180    g 47 77.45 44.811 11.63 58 176 176
r 40 75.98 45.681 11.95 70 176 180    g 43 81.19 49.091 10.85 64 162 170
g 44 81.42 39.442 13.08 63 174 176    b 38 81.87 60.055  8.63 48 170 186
b 44 73.03 50.541 10.13 45 168 168    r 45 87.66 37.388 14.03 56 186 192
b 45 66.45 44.754 11.12 51 176 176    r 47 79.15 47.273 10.60 47 162 164
g 54 83.12 51.855 10.33 50 166 170    b 49 81.42 49.156  8.95 44 180 185
g 51 69.63 40.836 10.95 57 168 172    b 51 77.91 46.672 10.00 48 162 168
r 48 91.63 46.774 10.25 48 162 164    g 49 73.37 50.388 10.08 67 168 168
r 57 73.37 39.407 12.63 58 174 176    g 54 79.38 46.080 11.17 62 156 165
g 52 76.32 45.441  9.63 48 164 166    b 50 70.87 54.625  8.92 48 146 155
g 51 67.25 45.118 11.08 48 172 172    b 54 91.63 39.203 12.88 44 168 172
b 51 73.71 45.790 10.47 59 186 188    r 57 59.08 50.545  9.93 49 148 155
b 49 76.32 48.673  9.40 56 186 188    r 48 61.24 47.920 11.50 52 170 176
```

```
b        52        82.78        47.467        10.50        53        170        172
          ;
```

When your program works correctly, paste it into the box below.

ANSWER 6

```
data Ggroup;
   input Group $  @; /* hold the data */
   if Group='g';  /* Sub-setting if statement */
                  /* only continue if condition true */
   input          /* code only reached if Team='red’ */
            Age
            Weight
            Oxygen
            RunTime
            RestPulse
            RunPulse
            MaxPulse
            @@    /*more data on the same line*/
            ;
   datalines;
r 44 89.47 44.609 11.37 62 178 182   g 40 75.07 45.313 10.07 62 185 185
g 44 85.84 54.297  8.65 45 156 168   b 42 68.15 59.571  8.17 40 166 172
r 38 89.02 49.874  9.22 55 178 180   g 47 77.45 44.811 11.63 58 176 176
r 40 75.98 45.681 11.95 70 176 180   g 43 81.19 49.091 10.85 64 162 170
g 44 81.42 39.442 13.08 63 174 176   b 38 81.87 60.055  8.63 48 170 186
b 44 73.03 50.541 10.13 45 168 168   r 45 87.66 37.388 14.03 56 186 192
b 45 66.45 44.754 11.12 51 176 176   r 47 79.15 47.273 10.60 47 162 164
g 54 83.12 51.855 10.33 50 166 170   b 49 81.42 49.156  8.95 44 180 185
g 51 69.63 40.836 10.95 57 168 172   b 51 77.91 46.672 10.00 48 162 168
r 48 91.63 46.774 10.25 48 162 164   g 49 73.37 50.388 10.08 67 168 168
r 57 73.37 39.407 12.63 58 174 176   g 54 79.38 46.080 11.17 62 156 165
g 52 76.32 45.441  9.63 48 164 166   b 50 70.87 54.625  8.92 48 146 155
g 51 67.25 45.118 11.08 48 172 172   b 54 91.63 39.203 12.88 44 168 172
b 51 73.71 45.790 10.47 59 186 188   r 57 59.08 50.545  9.93 49 148 155
b 49 76.32 48.673  9.40 56 186 188   r 48 61.24 47.920 11.50 52 170 176
b 52 82.78 47.467 10.50 53 170 172
;
run;
proc print data = Ggroup;
run;
```

QUESTION 7
Two-dimensional data

Data subjects do not have to fit on one line of the input buffer. Free-format data commands allow you to set the location of the raw data in 2 dimensions.

Self assessment question

Download the file: EXAMPLE2D.DAT and read the following data from the file:

Line	**Column**	**Variable**	**Type**
1	1-5	id	numeric

1	6-7	line	numeric
1	8-15	v1	numeric
2	19-21	v2	numeric
2	58-59	v3	character
2	79-83	v4	character
3	20	v5	numeric
11	20-14	v6	numeric

Paste your code into the box below when it works correctly.

Hints:

1. Each observation is 11 lines long.
2. The input buffer should be at least 100 characters in size, in order to hold the raw data for each line - see LRECL.
3. The slide set summarizes the positioning commands available to read the variables.

ANSWER 7

```
filename pwd "C:\Users\Folashikemi\OneDrive - De Montfort
University\P2586104 (vfiler1.lec-admin.dmu.ac.ukHome4)\Documents\IMAT5168
Analytical Programming Mark\Lab 4";
data Example2D;
infile pwd(Example2d.dat);
      input
      /*    line  pos         variable    format */
            #1          @1          id          5.
                        @6          line        2.
                        @8          v1                8.
            #2          @19         v2                3.
                        @58         v3 $        2.
                        @79         v4 $        5.
            #3          @20         v5                1.
            #11         @20         v6                4.
      ;
run;
proc print data = Example2D;
run;
```