## **CP5805 – Assessment 3: Data Analysis – Design**

**main()**:this function will load and display the main menu to the user and will store any data loaded into the program by the user. When the user selects a valid menu choice by selecting an integer from one to six, this function will call the appropriate sub-function to action that choice. Once the program has run through all sub-functions corresponding to that menu choice, the program will return the user to the main menu.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| menu\_choice | filename  dataset  integer  new\_name  sorted\_set  statistical\_report | main\_menu  data\_display  statistical\_report |

function main()

display main\_menu

get menu\_choice, displaying prompt

while menu\_choice != 6

if menu\_choice == 1

call get\_filename(prompt)

then call get\_data(filename)

otherwise if menu\_choice == 2

call check\_for\_loaded\_data()

display dataset

otherwise if menu\_choice == 3

call check\_for\_loaded\_data()

call get\_set\_choice(prompt, minimum, maximum\_number\_of\_sets)

then call rename\_a\_set(integer)

otherwise if menu\_choice == 4

call check\_for\_loaded\_data()

call get\_set\_choice(prompt, minimum, maximum\_number\_of\_sets)

then call sort\_a\_set(integer)

otherwise if menu\_choice == 5

call check\_for\_loaded\_data()

call get\_set\_choice(prompt, minimum, maximum\_number\_of\_sets)

then call generate\_statistical\_report(integer)

otherwise

display “Invalid option. Please select a value between 1 and 6.”

display main\_menu

get menu\_choice, displaying prompt

**get\_filename(prompt)**: this function is called by *main()* when the user selects the first option “Load data from a file” by entering the number one. It prompts the user to supply a filename, which is expected to be a CSV file in the same directory as the program (the working directory). If the user supplies a filename that does not match anything in the directory or is in the incorrect format, this function will display an error message to the user requesting that he/she enter another filename.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| prompt |  | filename |

function get\_filename(prompt)

get filename, displaying prompt

while filename cannot be found in this directory OR filename is not a CSV file

display “This file cannot be found. Please enter a valid filename.”

get filename, displaying prompt

return filename

**get\_data(filename)**: this function uses the filename supplied by *get\_filename()* to retrieve the data contained in that file. This data is then stored in *main()* as a list of lists for use in other program functions.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| filename | datafile  set\_name  set\_values | list\_of\_lists |

function get\_data(filename)

get filename

load each set as its own list, separating elements by commas

for each set in datafile

set\_name = set[0]

set\_values = set[1 through to end of set],

return list\_of\_lists

**check\_for\_loaded\_data**(): this function checks if a dataset has been loaded into the program when the user selects the second through fifth options from the main menu (“Display the data to the screen”, “Rename a set”, “Sort a set” and “Analyse a set” respectively) by entering the numbers two, three, four or five. If the user tries to select one of these menu option without a loaded dataset, this function will display an error message to the user and return to the main menu.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
|  | list\_of\_lists |  |

function check\_for\_loaded\_data()

if no list\_of\_lists exists in this directory

display “There is no data available.”

return to main\_menu

otherwise

return “Data available.”

**get\_set\_choice(prompt, minimum, maximum\_number\_of\_sets)**: this function will be called by *main()* via *check\_for\_loaded\_data()* when the user selects the third through fifth options from the main menu (“Rename a set”, “Sort a set” and “Analyse a set” respectively) by entering the numbers three, four or five. It prompts the user to enter an integer less than or equal to the number of sets in the current *dataset*. If the user tries to select a set that is unavailable by entering any other integer value, this function will display an error message to the user.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| prompt  minimum  maximum\_number\_of\_sets |  | integer |

function get\_set\_choice(prompt, minimum, maximum\_number\_of\_sets)

if check\_for\_loaded\_data() == “Data available”

get integer, displaying prompt

while integer < minimum OR integer > maximum\_number\_of\_sets

display error message

get integer, displaying prompt

return integer

**rename\_a\_set(integer)**: this function is called by *main()* via *get\_set\_choice()* when the user enters an integer corresponding to the “Rename a set” option on the main menu. It replaces the current name of a specified set with a new, valid name and displays a confirmation message to the user.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| integer | set\_name  current\_name  new\_name |  |

function rename\_a\_set(integer)

get integer

get set\_name of set

current\_name = set\_name

if proposed\_name is valid

replace current\_name with new\_ name AND display “*current\_name*

“renamed to ” *new\_name*”

otherwise

return to main menu

**validate\_proposed\_name(proposed\_name)**: this function is called by *rename\_a\_set()*. If *proposed\_name* is a valid name according to specific rules set within the program then *new\_name* is passed back to *rename\_a\_set()*.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| proposed\_name | list\_of\_lists\_names | new\_name |

function validate\_proposed\_name(proposed\_name)

get proposed\_name

if proposed\_name is an empty string OR proposed\_name already exists in list\_of\_lists\_names

display “Invalid set name. Please try again.”

otherwise

proposed\_name = new\_name

return new\_name

**sort\_set(integer)**: this function is called by *main()* via *get\_set\_choice()* when the user enters an integer corresponding to the “Sort a set” option of the main menu. It sorts the values in the specified set in ascending order and returns *sorted\_set* for later use by *main()*.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| integer | list\_of\_lists  set\_to\_be\_sorted | sorted\_set |

function sort\_set(integer)

get integer

set\_to\_be\_sorted = set[set\_values]

sorted\_set = sort set\_to\_be\_sorted in ascending order

return sorted\_set

**generate\_statistical\_report(integer)**: this function is called by *main()* via *get\_set\_choice()* when the user enters an integer corresponding to the “Analyse a set” option of the main menu. It calculates five different statistics for the specified set and compiles them back into a report that is passed back to *main()*.

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| integer | list\_of\_lists  sorted\_set  number\_of\_values  minimum  maximum  median  mode | statistical\_report |

function generate\_statistical\_report(integer)

get integer

number\_of\_values = length of set

minimum = min(set\_values)

maximum = max(set\_values)

if number\_of\_values is even

median = 0.5 \* (sorted\_set[number\_of\_values / 2] +

sorted\_set[number\_of\_values / 2 – 1])

otherwise

median = sorted\_set[number\_of\_values / 2]

for each unique value in dataset[integer]

count the number of occurrences

save the values with the maximum number of occurrences in mode\_list

if length of mode\_list >= 2

mode = “no unique mode”

otherwise

mode = max(mode\_list)

return statistical\_report