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
```