Monash College

MCD4720 - Fundamentals of C++

Assignment 1 - Trimester 2, 2022

Submission guidelines

This is an individual assignment, group work is not permitted

Deadline: July 22, 2022, 11:55pm (Week 5)

Weighting: 10% of your final mark for the unit

Late submission:

- By submitting a Special Consideration Form or visit this link: https://lms.monashcollege.edu.au/course/view.php?id=1331

- Or, without special consideration, you lose 5% per day that you submit late (including weekends). Submissions will not be accepted more than 14 days late. This means that if you got Y marks, only (Y - Y×n×0.05) will be counted where n is the number of days you submit late.
- Assessment items will not be accepted after more than 14 calendar days unless a Special Consideration application has been approved. This 14-day time frame does not apply to assessments due in Week 12.

Marks: This assignment will be marked out of 100 points, and count for 10% of your total unit marks.

Plagiarism: It is an academic requirement that the work you submit be original. If there is any evidence of copying (including from online sources without proper attribution), collaboration, pasting from websites or textbooks, Zero marks may be awarded for the whole assignment, the unit or you may be suspended or excluded from your course. Monash Colleges policies on plagiarism, collusion, and cheating are available here or see this link:

https://www.monashcollege.edu.au/__data/assets/pdf_file/0010/17101/dip-assessment-policy.pdf

Further Note: When you are asked to use Internet resources to answer a question, this does not mean copy-pasting text from websites. Write answers in your own words such that your understanding of the answer is evident. Acknowledge any sources by citing them.

Submission Instructions:

Your project must be submitted as a CLion project, including all header and code files, and any appropriate text files to ensure the program compiles and runs.

You may complete the tasks in your preferred IDE, however you **MUST** create a CLion in order to submit. Your project folder must be identified by using your name and assignment number, such as **YourFistNameLastName_A1.**

The entire project folder must then be zipped up into one zip file for submission. The zipped file **MUST** be named "**YourFistNameLastName_A1.zip**". This zip file must be submitted via the Moodle assignment submission page.

Explicit assessment criteria are provided, however please note you will also be assessed on the following broad criteria:

- ✓ Meeting functional requirements as described in the exercise description.
- ✓ Demonstrating a solid understanding of object-oriented design and C++ coding, including good practice.
- ✓ Following the Unit Programming Style Guide.
- ✓ Creating solutions that are as efficient and extensible as possible.

NOTE! ***Your submitted program MUST compile and run. Any submission that does not compile will automatically be awarded a 50 marks penalty. This means it is your responsibility to continually compile and test your code as you build it.***

NOTE! Your submitted files must be correctly identified and submitted (as described above). Any submission that does not comply will receive an automatic 20 marks penalty (applied after marking). Your CLion project should include all .cpp, .h and .txt files.

NOTE! Your tutor may ask you to program part of your assignment in class to determine that you have completed the work yourself. Failure to do this to an acceptable level will result in you being referred to the Subject Leader for plagiarism.

Scenario:

You have just joined a new multimedia company called MonashSchool. As a new employee, you have been tasked with creating a text-based prototype for a new maths tool the business has been asked to create by a major client called *"The Results Statistics generator"*.

Your job is to demonstrate some of the basic functionality of the program, which will be further developed by the team of programmers after you finish the prototype. To do this you will complete a series of tasks, each building upon the previous task.

Task List:

- Your project must include an application file – correctly formatted including variable and function declarations
- Create and display a menu that accesses each of the following functions of the program.
- Read data from a file and display it as an information screen about the application.
- Read some sample data from a file and store it in appropriate variables.
- Generate a statistical pattern using the stored data and display the result.
- Save new statistical pattern to file, at the user's request, appending each new pattern saved.
- Load all saved data from the saved file (if required) and display the contents.

Task 1: Create and Display a Menu

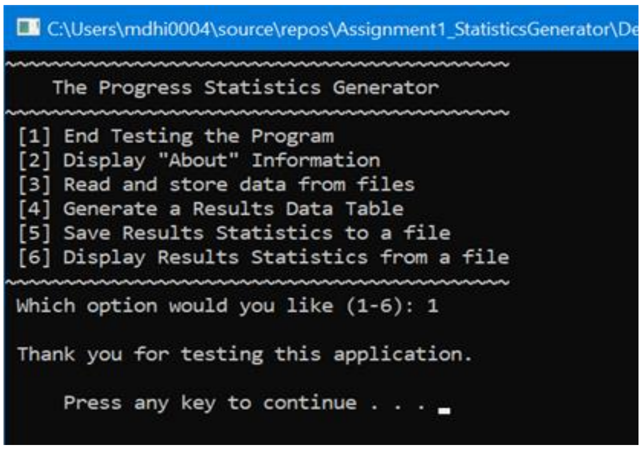
Your first task is to create and display a menu, just like the one shown here. In the example, option [1] has been selected, terminating the program.

Create a function called **`runMenu()`** and call it from the **`main()`** function.

From this function, the user must be able to select any of the displayed options as often as they want before ending the program.

Ensure that you validate the user's input so that only the options displayed can be chosen.

You should process the options as numeric such as `[1]` for ending the program, `[2]` for displaying the information, etc.


```
C:\Users\mdhi0004\source\repos\Assignment1_StatisticsGenerator\De
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    The Progress Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[1] End Testing the Program
[2] Display "About" Information
[3] Read and store data from files
[4] Generate a Results Data Table
[5] Save Results Statistics to a file
[6] Display Results Statistics from a file
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Which option would you like (1-6): 1

Thank you for testing this application.

    Press any key to continue . . . _
```

Task 2: Display an Information Screen

Your next task is to display some information about the application, just like the one shown here.

Create a function called **`displayText()`** and call it from the **`runMenu()`** function, when option **[2] Display "About" Information** is selected.

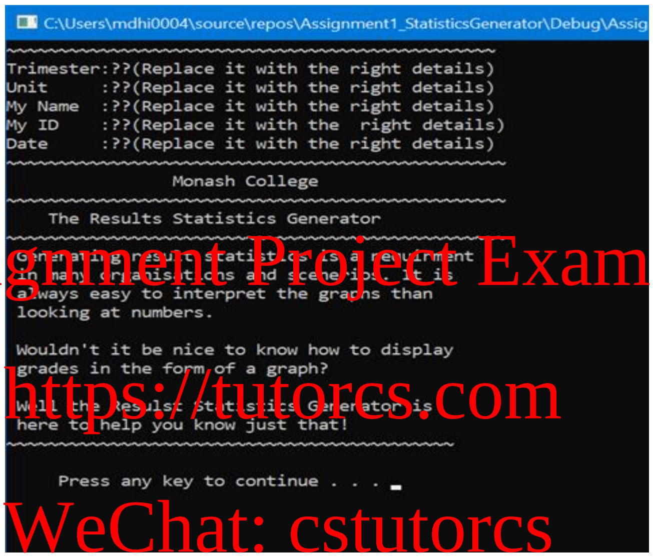
This function must accept a filename as a parameter ("About.txt") and successfully read and display the data.

You must also allow the user some time to read the displayed information before returning them to the menu.

You can copy **About.txt** into the appropriate folder to read and display, instead of creating your own. However, you may need to modify the format of the data, depending on how you intend to import it into your program.

You may create your own information, if you wish. If you do, keep it short!

Update the file to add your details (such as trimester, unit, …)


C:\Users\mdhi0004\source\repos\Assignment1_StatisticsGenerator\Debug\Assig

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Trimester:??(Replace it with the right details)
Unit     :??(Replace it with the right details)
My Name  :??(Replace it with the right details)
My ID    :??(Replace it with the  right details)
Date     :??(Replace it with the right details)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Monash College
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    The Results Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Generating result statistics is a requirment
 in many organisations and scenerios. It is
 always easy to interpret the graphs than
 looking at numbers.

 Wouldn't it be nice to know how to display
 grades in the form of a graph?

 Well the Resulst Statistics Generator is
 here to help you know just that!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

     Press any key to continue . . .
```

Task 3: Read and Store Data from a File

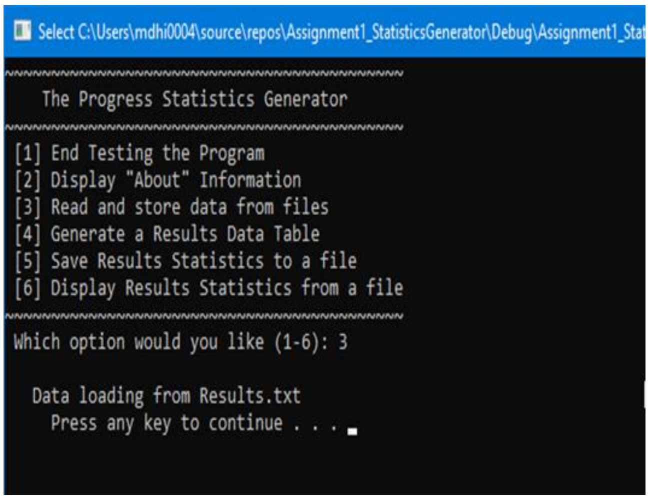
Your next task is to read and store data elements, from a text file which contains marks for 100 students in 10 assessments. And store this data in appropriate variables. Create a function called **`createLists()`** and call it from the **`runMenu()`** function, when option **[3] Read and Store Data from File** is selected.

This function must accept a filename as a parameter ("*.txt") and successfully read and store the data. The data is to be stored in a collection variable named **`progressData`** (collection variable means array or vector).

Apply appropriate data validation and display suitable error messages where required.

You may also be required to manipulate the string data, using appropriate string methods when displaying it.

Note: read the provided file (Results.txt) with results of 100 students for 10 assessments.



```
Select C:\Users\mdhi0004\source\repos\Assignment1_StatisticsGenerator\Debug\Assignment1_Stat
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   The Progress Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 [1] End Testing the Program
 [2] Display "About" Information
 [3] Read and store data from files
 [4] Generate a Results Data Table
 [5] Save Results Statistics to a file
 [6] Display Results Statistics from a file
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Which option would you like (1-6): 3

   Data loading from Results.txt
     Press any key to continue . . .
```

Task 4: Generate and Display a Statistics Pattern

Your next task is to generate a Statistics Pattern and display the result as a concatenated or formatted string.

Create a function called **`generateTable()`** and call it from the **`runMenu()`** function, when **[4] Generate a Statistics Pattern** is selected.

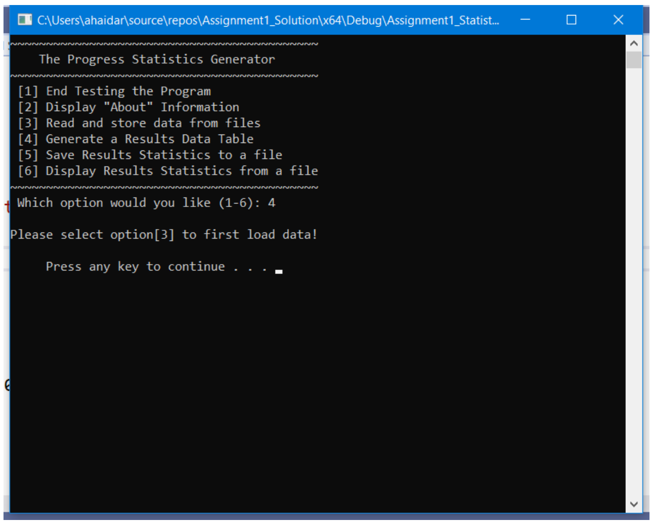
This function must return a concatenated or formatted string using input from the user (answering the question to set the parameters for number of students in class) and use the answer to generate the contents of the table. **Note:** the file you read contains marks for 100 students. You can choose any number between 1-100 to generate a statistics pattern for a definite number. The file contains marks for 10 different assessments. Users can generate a random number to pick one of the assessments to generate a statistics table.

You will need to apply a few different string manipulation techniques to achieve the desired result. Think through the process carefully before you start to write your code. **Hint:** work it out using pen and paper first.

Ensure that the collection variable (**`progressData`**) has content before trying to select each element, to stop the program from crashing.

This variable must be reusable, so also think of ways in which to reset the data so that any subsequent Tables generated are not contaminated by old data.

You can break this function into smaller sub-functions, if you find this will make the coding easier for you. Your marks will not be affected either way.

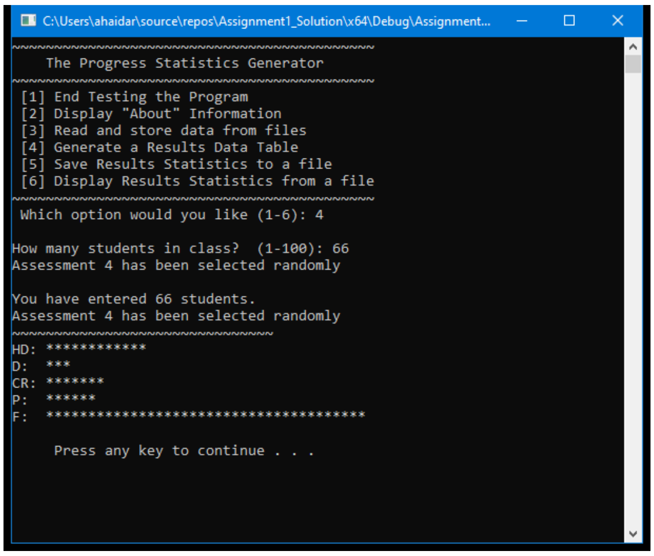


```
C:\Users\ahaidar\source\repos\Assignment1_Solution\x64\Debug\Assignment1_Statist...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   The Progress Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 [1] End Testing the Program
 [2] Display "About" Information
 [3] Read and store data from files
 [4] Generate a Results Data Table
 [5] Save Results Statistics to a file
 [6] Display Results Statistics from a file
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Which option would you like (1-6): 4

Please select option[3] to first load data!

     Press any key to continue . . .
```

If option 4 has been selected before option 3



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    The Progress Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 [1] End Testing the Program
 [2] Display "About" Information
 [3] Read and store data from files
 [4] Generate a Results Data Table
 [5] Save Results Statistics to a file
 [6] Display Results Statistics from a file
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Which option would you like (1-6): 4

How many students in class?  (1-100): 66
Assessment 4 has been selected randomly

You have entered 66 students.
Assessment 4 has been selected randomly
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
HD: ************
D:  ***
CR: *******
P:  ******
F:  **************************************

     Press any key to continue . . .
```

See screenshot for an example of the required output.

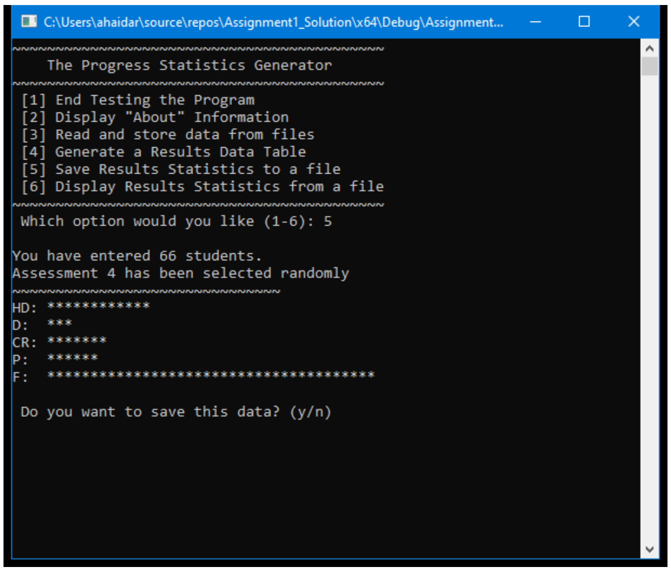
Task 5: Save the Statistics Table to a File

Your next task is to save the Statistics Table to a file, called "**savedProgressData.txt**". In this task you must:

- display the last generated Statistics Table
- ask the user if they want to save it
  - if so, add it to any existing data in the file
  - if not, do nothing
- return to the menu when the task is complete

Create a function called **`saveData()`** and call it from the **`runMenu()`** function, when option **[5] Save a Character to file** is selected.

Use the screenshot as a guide for your display and input request. You should also inform the user when the data is saved, before returning to the menu.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    The Progress Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 [1] End Testing the Program
 [2] Display "About" Information
 [3] Read and store data from files
 [4] Generate a Results Data Table
 [5] Save Results Statistics to a file
 [6] Display Results Statistics from a file
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Which option would you like (1-6): 5

You have entered 66 students.
Assessment 4 has been selected randomly
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
HD: ************
D:  ***
CR: *******
P:  ******
F:  **************************************

 Do you want to save this data? (y/n)
```

Task 6: Display Tally Statistics from a File

Your final task is loading the saved Statistic Tables from the "**savedProgressData.txt**" and displaying them appropriately.

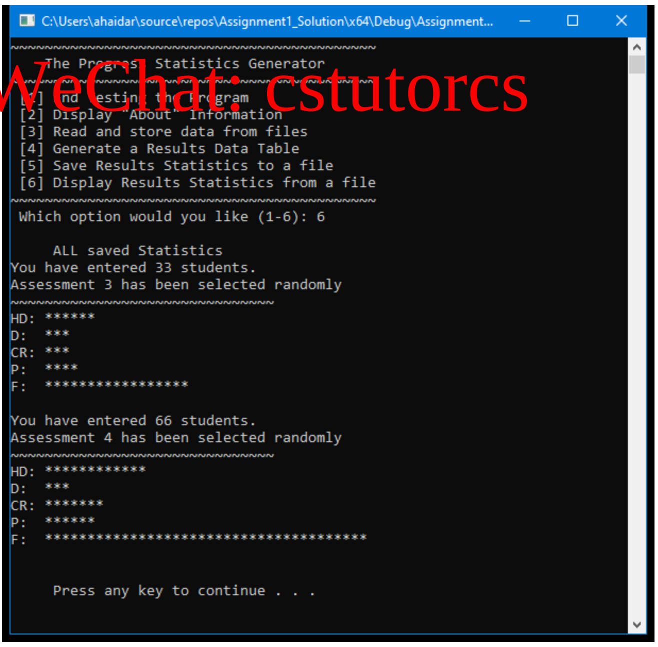
How you achieve this is dependent upon how you saved the data. Ideally, the display should be the same as the way you displayed the Statistics Table when you generated it.

Use the screenshot as a guide for your display – these Statistics Tables are being displayed in the same way as they were originally generated.

You must also load this data into the `progressData` collection variable first, then display the saved data.

Things you may need to consider:

- Do the collection lists need to be initialised before you display the saved data?
- How have you saved the data?
  - Did you save the random numbers for each element type?
  - Did you save the completed strings?
  - Did you use another method?
- How will you display the data after it is read from the saved data file?
  - Which method is the most efficient way to display it?
- How will you ensure that the user has time to read the output before returning to the menu?


C:\Users\ahaidar\source\repos\Assignment1_Solution\x64\Debug\Assignment...

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    The Progress Statistics Generator
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 [1] End Testing the Program
 [2] Display "About" Information
 [3] Read and store data from files
 [4] Generate a Results Data Table
 [5] Save Results Statistics to a file
 [6] Display Results Statistics from a file
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Which option would you like (1-6): 6

     ALL saved Statistics
You have entered 33 students.
Assessment 3 has been selected randomly
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
HD: ******
D:  ***
CR: ***
P:  ****
F:  *****************

You have entered 66 students.
Assessment 4 has been selected randomly
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
HD: ************
D:  ***
CR: *******
P:  ******
F:  **************************************


     Press any key to continue . . .
```

Assignment Notes:

It is your responsibility to know what is expected of you. If you are unsure about anything, ask your tutor sooner rather than later. Write any questions and/or notes here to help you clarify what you must do.

Assignment 1: Marking Criteria [100 marks in total]

> ***NOTE:*** *Your submitted project must be correctly identified and submitted, otherwise it will receive an automatic* ***20-mark penalty*** *(applied after marking).*

Does the program compile and run? Yes or No

> **NOTE!** *Your submitted program MUST compile and run. Any submission that does not compile will receive an automatic* ***50 marks penalty*** *(applied after marking).*

Marking Guide:

1. Application Design [20]
   - 1.1. Application File **[20]**
     - 1.1.1. The main() function has appropriate function calls to keep it uncluttered [4]
     - 1.1.2. Has all the correct #include statements [4]
     - 1.1.3. Has the required data members and member functions using meaningful names [4]
     - 1.1.4. Has created all function prototypes correctly [4]
     - 1.1.5. Appropriate additional functions created to streamline code [4]

2. Functionality **[60]**
   - 2.1. Task 1: Menu [12]
     - 2.1.1. The runMenu() function correctly initialises any required variables [2]
     - 2.1.2. The menu options are clearly displayed [2]
     - 2.1.3. Appropriate use of loop and decision structures to process user input [2]
     - 2.1.4. Correct use of function calls when responding to user input [2]
     - 2.1.5. The user controls when to return to the menu [2]
     - 2.1.6. Appropriate exit [2]

   - 2.2. Task 2: About Information **[8]**
     - 2.2.1. The displayText() function uses the correct argument [2]
     - 2.2.2. Appropriate validation checks have been made [2]
     - 2.2.3. Appropriate use of loop and decision structures to read and display data [3]
     - 2.2.4. Appropriately updated the file and added details [1]

   - 2.3. Task 3: Reading and Storing Data **[10]**
     - 2.3.1. The createLists() function uses the correct argument [2]
     - 2.3.2. Appropriate validation checks have been made [2]
     - 2.3.3. The collection variables (arrays or vectors) are correctly initialised [2]
     - 2.3.4. Appropriate method(s) used to read the data from the file [2]
     - 2.3.5. Appropriate use of loop and decision structures to read and store data [2]

   - **2.4.** Task 4: Generate and Display Statistics chart **[10]**
     - 2.4.1. The generateTable() function returns a string [2]
     - 2.4.2. Validation checks that collection lists have been created [2]
     - 2.4.3. The generated output clearly represents the data read [4]
     - 2.4.4. The returned string is correctly concatenated and/or formatted [2]

   - 2.5. Task 5: Save Data to a File **[12]**
     - 2.5.1. The saveData() function uses the correct argument(s) [2]
     - 2.5.2. The current statistics chart is displayed appropriately [2]
     - 2.5.3. The user is asked whether to save or not with an appropriate process of user input [2]
     - 2.5.4. Appropriate validation for writing the file is in place [2]
     - 2.5.5. The string saved into the specified file [2]
     - 2.5.6. The data is correctly added to the specified file [2]

   - 2.6. Task 6: Load Data from a File **[8]**
     - 2.6.1. The loadData() function uses the correct argument(s) [2]
     - 2.6.2. Appropriate validation checks, loop and decision structures to read and display data [2]
     - 2.6.3. Generated data is displayed using correct formatting after loading [4]

3. Quality of Solution and Code **[20]**
    3.1. Does the program perform the functionality in an efficient and extensible manner? [4]
    3.2. Has a well-designed program been implemented? **[8]**
        3.2.1. Demonstrates the use of logical procedures [4]
        3.2.2. Code is indented correctly [2]
        3.2.3. White spaces are used properly [2]

    3.3. Has the Programming Style Guide been followed appropriately? **[8]**
        3.3.1. All functions and variables have meaningful names and use correct camel notation [4]
        3.3.2. Appropriate use of comments throughout the code **[4]**
            3.3.2.1. The top of any program file [1]
            3.3.2.2. Above every function [2]
            3.3.2.3. In line [1]