CS 135 - Project 3 -- Array Processing Program

INTRODUCTION

The purpose of Project 3 is to demonstrate all the skills that you have learned in Chapters 1-8 of the textbook. In particular: File I/O (Chapter 5); Functions and modular design (Chapter 6); Arrays and Vectors (Chapter 7); Searching and Sorting Arrays (Chapter 8).

INSTRUCTIONS

The program shall read an input file of whole numbers into an array, compute statistical values based on those numbers, and write the results to the screen and to an output file.

Write a program to:

(1) read a set of whole numbers into an array
(2) compute the minimum value
(3) compute the maximum value
(4) compute the total of the numbers
(5) compute the average
(6) compute the standard deviation
(7) compute the median
(8) compute the mode(s)
(9) write the results to the terminal screen and to an output file
(10) draw a histogram for the set of numbers (Optional, for extra credit)

When you are finished and want to test your program, document your tests by taking screenshots:
Use *Alt+PrtScn* to copy the active window, and paste it into an MS-Word document.
When making the Word doc, include a description for each screenshot that tells what is going on in it:
what the test was; what the result was; etc.
The Word doc should include at least 5 screenshots, along with associated description.
Copy the MS-Word document to the ***Proj3*** folder that you will create for this project.

Data Types

The program shall use these data types.

DATA	TYPE
Raw data	array of integer type
Minimum	integer type
Maximum	integer type
Total	integer type
Average	floating point type
Standard deviation	floating point type
Median	floating point type
Mode	integer type

Output File

The program shall produce a formatted output file with this information:

(1) Numbers of data points
(2) Minimum
(3) Maximum
(4) Total
(5) Average
(6) Standard deviation
(7) Median
(8) Mode(s)

PROGRAM REQUIREMENTS

An implied requirement of the program is: to compute the median and mode, the data must be sorted.
"Sorted" means the values must be arranged in ascending order (from smallest to largest).
The program shall include a separate function to sort the data. Sorting algorithms are discussed in the textbook.
The program shall include separate functions for each of the 10 items listed above under "Instructions".
The program shall find the minimum and maximum values before the data are sorted.
If you are statistics geek, and need to ask, assume that the data set is population data, and not sample data.

An in-class demonstration is required for this project. This will be on-line, over the Internet.
Not giving a demo to the class will result in a grade of 0.

Program Design

The programmer is responsible for designing and implementing a user interface, using menus and prompts.
The user must be able to specify the name of the input file and the output file.
For this program, you can assume that no more than 1000 numbers will be in any input file.
Since it is possible to have multiple modes, and it is not possible to know how many there will be in advance, the modes should be stored in a vector.

Here is a rough program design, which you will need to flesh out.

STEP 1. Get name of input file from user.
STEP 2. Open input file.
STEP 3. Read input file into an array (the array's size must be at least 1000).
STEP 4. Close input file.
STEP 5. Compute the values:
---- Minimum
---- Maximum
---- Total
---- Average
---- Standard deviation
STEP 6. Sort the array (smallest-to-largest) and find:
---- Median
---- Mode(s)
STEP 7. Write results to screen.
STEP 8. Get name of output file from user.
STEP 9. Open output file.
STEP 10. Write results to output file.
STEP 11. Close output file.
STEP 12. Terminate program.

CODING CONVENTION

- Each file shall have a file header.
- Each function shall have a function header.
- The code shall be properly indented and commented.
- The code shall include the lines in the *pgm_template.cpp* at the end of the *main()* function that will output your name and the compilation date, and pause the program.
- All detailed code shall reside in functions other than *main()*, and *main()* shall not be a "wrapper" function.
- The program shall be written using procedural programming and modular design. (No "goto's" or OOP.)
- No global variables are to be used. Data must be passed using the techniques of Chapter 6.
- Data shall be stored in integer ***arrays***. Modes shall be stored in ***vectors***.
- Function recursion shall not be used. All loops must be one of the three types discussed in Chapter 5.

PROJECT GRADING

1) Basics:
   (a) Is the project complete?
   (b) Is the code correctly compiled?
   (c) Does the program run by double-clicking, without crashing?
2) Did the student follow the written instructions?
   It is invalid to reinterpret or change the instructions in order to redesign or simplify the Project.

SUBMISSION INSTRUCTIONS

Compile and test your code in the MinGW environment we have at TMCC. That is how it will be graded.

Create a folder named ***Proj3*** containing the following folders and contents:
Array_Processor, containing:
- *Array_Processor.exe*
- *Array_Processor.cpp*
- *.txt* files (input and output files)

The ***screenshots.docx*** file shall be in the main folder: ***Proj3***.

Zip up your folder and submit your ***Proj3.zip*** file to the Canvas drop box.