Programming Assignment #6A: Using Arrays and File Input/Output

Specification: This program can help you get the general idea of how software such as Microsoft WORD is implemented. Your program should provide services for the user to **enter** test scores from the **keyboard** and store the scores in an array, to **write** (**save**) the scores currently stored in the array into a given file, to **read** a collection of test scores that are already stored in a given file into an array, to **display** the scores currently stored in the array and to calculate and display the related **statistics** of the scores.

Data Structures (variables and the array) to retain the information in the main memory of the computer:

- You should declare in your main function a variable *int numStudents* to dynamically store the information of the actual number of students involved. This allows the user to adapt the number of scores to process based on the current number of students. In the beginning, you should initialize the value of this variable to 0.
- You should also declare an array *double scores[100]* in your main function for storing the scores one by one starting from the beginning of this array toward the end of the array, up to 100 scores at most.
- In the main function, you should declare a string variable *string filename* to store the information of a file name entered by the user. You should also declare two file handles *ifstream fin;* and *ofstream fout;* for file input/output purposes respectively

File format to retain the information in files: A file for holding the information of test scores for Programming #6A should consist of only numbers with each number on a separate line. The first number in the file should be the number of students in the class, followed by the scores of the students one by one. For e**xample**, the following would be the contents in a file showing the scores of **three students** in a class:

3
95
77
88

Service menu and the analogy to Microsoft WORD: Like the ATM service program in Programming #4A, the main function should set up a loop to repeatedly display a menu to the user, prompt the user to choose one of the options below, and then provide corresponding service.

- ***K***: This option allows the user to use the ***K***eyboard to manually reset and re-enter the information of (i) the number of students in the class stored in *numStudents* and (ii) the corresponding number of scores to be stored in the array *scores[100]* accordingly.
  Analogy to Microsoft WORD: This option is just like doing ***File*** ➔ ***New*** under WORD to create new contents about the number of students and the test scores and temporarily store the information in the variable *numStudents* and the *scores* array and in the main memory.

- ***W***: This option allows the user to ***W***rite into a file (i) the value of *numStudents*, i.e. the number of students in the class and (ii) the test scores currently stored in the array *scores[100]*.
  Analogy to Microsoft WORD: This option is just like ***File*** ➔ ***Save*** *As* under WORD to permanently save the contents in the *scores* array and the variable *numStudents* from the main memory into a file.

- ***R***: This option allows the user to (i) ***R***ead from a file a new value of *numStudents* i.e. the number of students in the class, and then (ii) ***R***ead the corresponding number of test scores from the file and store them in the array *scores[100]*.
  Analogy to Microsoft WORD: This option is just like ***File*** ➔ ***Open*** under WORD to read existing the contents stored in file into the *scores* array and the variable *numStudents* in the main memory.

- ***D***: This option allows the user to ***D***isplay the number of students and the list of all the scores currently stored in the array, and in addition calculate and display the **average** of the scores, the **highest** score, and the **lowest** score.
  Analogy to Microsoft WORD: This option is just like ***Review*** ➔ ***Word Count*** under WORD that allows you to review the basic statistics regarding the contents stored in the *scores* array and the variable *numStudents* in the main memory.

- ***M***: This option allows the user to ***M***odify one of the scores by re-entering the score from the keyboard.
  Analogy to Microsoft WORD: This option is just like editing the current contents under WORD.

- **Q**: This option allows the user to ***Q***uit the program.
  Analogy to Microsoft WORD: This option is just like ***File*** ➔ ***Close*** under WORD to close the program.

Implementation of the services: On each iteration of that loop, the program should read the user's choice and use a *switch* statement to do things according to the choice:

- If the choice is '**K**': the program should ask the user to input the number of students and read it into *numStudents*, and the program should check to make sure this number is a non-negative integer less than or equal to 100. The program then should set up a loop to read in the scores from the keyboard according to the current number of students and store the scores one by one starting from the beginning of the *scores* array toward the end of that array.
- If the choice is '**W**': the program should ask the user to provide the name (less than 20 characters) of an output file (e.g. *scores.txt*), and read the file name into a string variable *filename*. The program should then open that file, **write *numStudents*, the number of students, into the file**, and then **set up a loop to write the scores currently stored in the array one by one into that file**. When writing these numbers to the file, remember you should also **separate these numbers by writing either a space or an *endl*** immediately after writing a number into the file.

- If the choice is '**R**': the program should ask the user to provide the name (less than 20 characters without spaces in it) of an input file (e.g. *scores.txt*), and read the file name into a string variable *filename.* The program should then open that file and read the first integer from the file into *numStudents*, and the program should check to make sure this number is a non-negative integer less than or equal to 100. The program should then set up a loop to read in these scores one by one from that file, and store the scores one by one into the array, starting from the beginning of the *scores* array toward the end of that array. In the end, the program closes the connection to that file.
- If the choice is '**D**': the program should display the current number of students and set up a loop to display the corresponding scores currently stored in the array, and in addition the program should set up loops to calculate the following ***S***tatistics: the average of the scores, the highest score, and the lowest score, and then display them on the screen too.
- If the choice is '**M**': the program should (i) ask the user the index of the score he/she wants to modify, (ii) read the index as an integer from the user into the a local variable, say, *i*, (iii) make sure the index is a valid one, and (iv) ask the user to input a new score from the keyboard to replace the old score stored in *scores[i]* in the array.
- If the choice is '**Q**', the program will output a thank-you message and then loop would finish to end the program.

Technical details about file input/output operations:

- You should have a line *#include <fstream>* in the beginning of your .cpp file to include the support of file input/output library.
- You should have a line *#include <string>* in the beginning of your .cpp file to include the support of string library.
- Read sections 5.11 of the textbook and the descriptions below to see (i) how to declare file stream objects as files handles to open files for input/output respectively and (ii) how to read from or write into files and how to close the files in the end.
- In the main function, you should declare a string variable *string filename* to store the file name given by the user. You should also declare two file handles *ifstream fin;* and *ofstream fout;* for file input and file output purposes respectively
- To read a file name from the user and store it in the file name string, you can use the statement *cin >> filename;*
- To open an input file whose file name is stored in the file name string, you can use the statement *fin.open(filename);*
- To read things from the input file, use *fin >> …;*
- To close the input file currently accessed from fin, use *fin.close( );*
- To open an output file whose file name is stored in the file name string, you can use the statement *fout.open(filename);*
- To write things to the output file, use *fout << …;*
- To close the output file currently accessed from fout, use *fout.close( );*