CSCI1300 - Introduction to Computer Programming
Instructor: Dr Knox
Due by Sunday 5PM

Recitation 8: File IO

During this class so far, we have been using the iostream standard library. This library provided us with methods like, cout and cin. *cin* method is for reading from standard input and *cout* is for writing to standard output.

In this recitation we will discuss *file Input/Output*, which will allow you to read and write from a file. In order to use these methods, we will need to use another c++ standard library, fstream.

Headers to include:

These headers must be included in your C++ file before you are able to process files.
#include <iostream>
#include <fstream>

Opening a file:

The first step to processing files is to open the file, with ***open(filename)***. It must be opened before you are allowed to read from it or write to it. In order to open a file, the *ofstream*, *fstream*, or *ifstream* objects should be used. If you want to open the file for writing, either the ofstream or fstream object may be used. If you want to open a file for reading only, then the ifstream object should be used.

For example:

```
ofstream myfile;
myfile.open("filename");
```

Variables as file names:

If you want to store the file-name into a string variable first and then use the variable name to open the file, in that case you will have to use the c_str() function.
The file name in an open function is a C-style string. Because of the specific restrictions of C++, you can't use a variable of type string as the parameter. However, if filename is

a variable of type string, you can obtain the corresponding C-style string by calling the function **filename.c_str()**, and you can use this function call as a parameter to open.
For example:

```
string filename;
ifstream datafile;
cout << "Please enter the input file name: ";
cin >> filename;
datafile.open( filename.c_str() );
```

Checking for open file:
It is always good practice to check if the file has been opened properly or send a message that it did not open properly. To check if a file stream successfully opened the file, you can use fileObject**.is_open().** This method will return a boolean value true or false.

You can also use the fileObject.**fail()** function to check if the file open was successful or not.

Reading from a file:

When you read from a file into your C++ program, you will use the stream insertion syntax you have seen with cin, which inputs information from the keyboard or user, **(>>)**. The difference is that you will be using the ifstream object instead of the cin object, allowing for the program to input or read from the file. Here you would be given a file with information to read. Then once the print statement is called, you will see the lines of the file printed to the terminal.

For example:

```
getline(myfile, line)
cout << line << endl;
```

Note: getline() will read the data in string format.

OR

```
myfile >> line;
cout << line<<endl;
```

Writing to a file:

When you write to a file from your C++ program, you will use the stream insertion syntax you have seen with cout, which outputs information to the terminal screen, **(<<)**. The difference is that you will be using the ofstream object instead of the cout object, allowing for the program to direct the output correctly.
For example:

```
myfile << “Writing this line to a file. \n”;
myfile << “Writing this second line to a file. \n”;
```

If you opened the text file, then you will see the string statements within the new output text file, such as:

Writing this line to a file.
Writing this second line to a file.

Closing a file:

When you are finished processing your files, it is always better to close all the opened files before the program is terminated. The standard syntax for closing your file is **close();** See the above example used in opening a file.

Recitation 6 exercise:
Write a program that reads the scores of a student from a file into an array and writes the average score and the corresponding grade in the output file. You can download the input file “Scores.txt” from moodle under recitation 8. You can name your output file as “Grade.txt”.
The input file contains scores of a student in 7 subjects with each score written on a newline. Your program must contain separate functions for reading and writing a file. You should declare the array of scores in your main function and pass it to these functions.

- Pass the input file name and the empty scores array to the readScores() function. This function should read the scores from the input file and store the scores into your array.

  void readScores(string inputFile , float scores)

Pass the output file name and the array of scores (which should be populated now) to the *writeGrade()* function. This function should calculate the average score of the student and write the average score and the corresponding grade to the output file in the following format: Average score, Grade

void writeGrade(string outputFile , float scores)
Use the following rules to calculate the grade.

For average scores greater than and equal to 90
    Grade is A
For average scores greater than and equal to 80 and lesser than 90)
    Grades is B
For average scores greater than and equal to 70 and lesser than 80)
    Grades is C
For average scores greater than and equal to 60 and lesser than 70)
    Grades is D
For average scores lesser than 60
    Grades is F

Note: Make sure your code is properly indented and commented.
Submit a zip file containing your .cpp program and the output text file under the link Recitation 8 Submit. Submissions are due Sunday 5pm.