CSCE 1030 Lab 10

General Guidelines: (for ALL of your programming assignments and labs)

- Use meaningful variable names.
- Use appropriate indentation.
- Use comments, especially for the header, variables, and blocks of code. *Please make sure that your submitted code has comments as points may be deducted for a lack of comments.*

Example Header:

```
/* Author:      Jane Doe (Jane.Doe@my.unt.edu)
   Date:
   Instructor:
   Description: A small description in your own words
                that describe what the program does. Any
                additional flags needed to allow the
                program to compile should also be placed
                here.
 */
```

Example Function Header:

```
/* Function:    Deposit
   Parameters:  a double representing account balance and
                a double representing the deposit amount
   Return:      a double representing account balance
                after the deposit
   Description: This function computes the account
                balance after a deposit.
 */
```

A. Member Functions get, put, fail, and eof

Create a file called **input10A.txt** and type (or copy) the following text exactly as it appears below into that file. You may cut and paste the following 7 blue lines (including the blank line between the two paragraphs) into that file:

C++ is a cross-platform language that can be used to create high-performance applications. C++ was developed by Bjarne Stroustrup, as an extension to the C language. C++ gives programmers a high level of control over system resources and memory.

C++ is one of the world's most popular programming languages. C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems. C++ is an

object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

The following program reads the entire contents of the **input10A.txt** file and attempts to display its entire contents exactly as it appears above:

```
// this program reads the entire contents of an input file and will
// display it with the same format.
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;

int main( )
{
    char c;
    ifstream in_s;  // declaration of the stream of type input

    char input_file[15];
    cout << "Enter the name of the input file: ";
    cin  >> input_file;

    in_s.open(input_file); // connect to the input file and test
    if (in_s.fail())
    {
        cout << "Unable to open input file " << input_file << endl;
        exit(EXIT_FAILURE);  // exit if cannot open file
    }

    cout<<"*** Here are the contents of the input file ***"<<endl;

    while (in_s>>c) // read all characters one-by-one to end of file
    {
        cout<<c;
    }

    cout<<endl<<"*** Done writing the contents of the file ***"<<endl;

    in_s.close();  // close the file

    return 0;
}
```

Compile and run the program, using the **input10A.txt** file as the input file. Did this program produce the same exact output as shown above? What do you think the problem is?

The problem is that the extraction operator does not read the white spaces, i.e., it skips blank spaces, tabs (\t), and new lines (\n). Thus, the entire text will appear in one piece without the separating spaces and new lines. In order to read and write the entire text with correct spacing, we will use a member function with the input stream. The get(c) function, where **c** is a character, allows us to read all characters from the file one character at a time.

So to fix the above program we could simply use this function instead of the extraction operator.
The member function eof() can be used with a stream of input type to determine the end-of-file. This function returns true when the end of the input file is reached. Thus, it can be used in a while loop to control the looping process. In general, you need to read one character before you check to see if the end of the file is reached.
Now, modify the above program by using the get() member function instead of the extraction operator (i.e., >>) as well as the eof() member function to read until the end of the file. But instead out displaying the contents to the terminal, modify the program so that it writes to whatever output file is specified by the user using the put() function.

To do this, you will need to modify the program so that it prompts the user for the filenames for two streams, one for the input and the other for the output. Note that you can remove the two cout statements in main with the "***" as these lines do not need to be written to the file. When testing, do not overwrite the input file **input10A.txt**, but instead use **out10A.txt**.

Complete the requested changes, and then save the file as **Lab10A.cpp**, making sure to compile and run the program to verify that it produces the correct results. Note that you will submit this file to Canvas.

B. Array Sorting

Create a file called **unsorted.dat** and copy (or type) the following floating point numbers into that file.

```
13.75
-29.89
15.25
28.74
-22.65
-23.78
-90.95
30.02
17.99
```

Now, create a program file called **Lab10B.cpp** with the following requirements.

- In your main function, read the **unsorted.dat** file in an array. You do not know exactly how many numbers there are in the file (your program should be generic), so declare an array of sufficiently large size and use the while loop to read the numbers into the array.

- Then, use a programmer-defined function to sort this array in ascending order. You cannot create a separate array for this purpose. You need to call this function and pass the array to this function from the main function.
- In your main function, write the sorted array to an output file named **sorted.dat**.

Complete the requested changes, and then save the file as **Lab10B.cpp**, making sure to compile and run the program to verify that it produces the correct results. Note that you will submit this file to Canvas.

C. Multiple Files

The following file is given to you: **data.dat**. Download it and take a look at it contents.
It has four data items in each line, separated by whitespaces – first name, last name, GPA and ID. Your objective is to copy every record whose GPA is less than 3.0 in the specified format. Your output file must look like: **selected.dat**. You **DON'T** need to submit either of the data files.
In your main function, you will open both input and output files using suitable file stream (ifstream and ofstream) operators and pass them to a function named **mycopy**. Write your main function in a separate file named **main.cpp**.

The function **mycopy** will accept the two arguments passed by the **main** function. Inside your function, read every line of the **data.dat** input file into suitable local variables (arrays are not needed) using the passed ifstream operator, one line at a time. Write the data to the **selected.dat** output file if the GPA is less than 3.0 using the passed ofstream operator.
You must write the data using the following formatting requirements:
Use 10 spaces for writing the first name.
Use 10 space for writing the last name.
Use 5 spaces for writing the GPA and there must a single digit after the decimal point.
Use 5 spaces for writing the ID.
You must use the I/O manipulators (NOT the stream member functions) for this purpose.

Save your function in its own separate file named **mycopy.cpp**.

Finally, create a header file named **my_header.h** and to include your header files and all other necessary declarations. Make sure to include your header file in your .cpp files.
Complete the requested changes, making sure to compile and run the program to verify that it produces the correct results. Note that you will submit your files to Canvas.

Now that you have completed this lab, it's time to turn in your results. Once you've moved the files to your windows machine (using **winscp**), you may use the browser to submit them to Canvas for the **Lab 10** dropbox.

You should submit the following files:

- **Lab10A.cpp**
- **Lab10B.cpp**
- **main.cpp**
- **mycopy.cpp**
- **my_header.h**

Ask your TA to check your results before submission. The above files *MUST* be submitted to Canvas by the end of your lab section.

Now that you've finished the lab, use any additional time to practice writing simple programs out of the textbook, lectures, or even ones you come up with on your own to gain some more experience.