ECE 4122/6122 Lab 4

(100 pts)

Section	Due Date
All Sections	Nov 7th, 2021 by 11:59 PM

Notes:

You can write, debug and test your code locally on your personal computer. However, the code you submit must compile and run correctly on the PACE-ICE server.

In this homework you will need to develop ***two*** executable programs. One program for the server and one for the client.

Grading Rubric

AUTOMATIC GRADING POINT DEDUCTIONS PER PROBLEM:

Element	Percentage Deduction	Details
Does Not Compile	40%	Code does not compile on PACE-ICE!
Does Not Match Output	10%-90%	The code compiles but does not produce correct outputs.
Clear Self-Documenting Coding Styles	10%-25%	This can include incorrect indentation, using unclear variable names, unclear/missing comments, or compiling with warnings. (See Appendix A)

LATE POLICY

Element	Percentage Deduction	Details
Late Deduction Function	score – 0.5*H	H = number of hours (ceiling function) passed deadline

TCP Sockets: Debug Logging Server

In real world applications debug/status messages from applications are often sent to a central logging server so that any issues that may occur can be investigated as to the cause. In this assignment you will be creating a TCP debug logging server.

Server-70 points

Write a console program that takes as a single **command line argument** the **port number** on which the **TCP Server** will listen for connection requests. Your server application needs to be able to maintain the connections from multiple clients. The clients will be sending text string messages that the server will save in a file called **server.log** in the same directory as the server application. The server will add a newline character to the end of each message when saving to the text file. All messages shall be **appended** to the server.log file if it already exists, otherwise a new file will be created. The server logs when clients connect and disconnect.

Client-30 points

You need to write a client console program that takes as a **command line argument** the **IP Address** and **port number** of the server as shown below:

./a.out localhost 61717

The program should continuously prompt the user for messages to send to the server.

Here is example prompting the user for messages

Please enter a message:

Testing your assignment:

1. Make sure that you test your server using multiple clients.
2. Make sure that your server can handle clients closing and reconnecting.
3. Make sure that both your server and client check the command line arguments for invalid entries:
    a. Negative port numbers
    b. Non numeric values
    c. Port number outside the range 61000-65535
    d. If an invalid command line is detected then the following message should be displayed showing the invalid value:
    Invalid command line argument detected: <invalid argument>
    Please check your values and press any key to end the program!

4. If the client application is not able to connect to the server, then client shall display the following error message:

   Failed to connect to the server at <IP address entered> on <port num>.
   Please check your values and press any key to end program!

Appendix A: Coding Standards

Indentation:

When using *if/for/while* statements, make sure you indent 4 spaces for the content inside those. Also make sure that you use spaces to make the code more readable.
For example:

```
for (int i; i < 10; i++)
{
    j = j + i;
}
```

If you have nested statements, you should use multiple indentions. Each { should be on its own line (like the *for* loop) If you have *else* or *else if* statements after your *if* statement, they should be on their own line.

```
for (int i; i < 10; i++)
{
   if (i < 5)
   {
       counter++;
       k -= i;
   }
   else
   {
       k +=1;
   }
   j += i;
}
```

Camel Case:

This naming convention has the first letter of the variable be lower case, and the first letter in each new word be capitalized (e.g. firstSecondThird). This applies for functions and member functions as well! The main exception to this is class names, where the first letter should also be capitalized.

Variable and Function Names:

Your variable and function names should be clear about what that variable or function is. Do not use one letter variables, but use abbreviations when it is appropriate (for example: “imag" instead of “imaginary”). The more descriptive your variable and function names are, the more readable your code will be. This is the idea behind self-documenting code.

File Headers:

Every file should have the following header at the top

```
/*
Author: <your name>
Class: ECE4122 or ECE6122
Last Date Modified: <date>

Description:

What is the purpose of this file?

*/
```

Code Comments:

1. Every function must have a comment section describing the purpose of the function, the input and output parameters, the return value (if any).
2. Every class must have a comment section to describe the purpose of the class.
3. Comments need to be placed inside of functions/loops to assist in the understanding of the flow of the code.