CSci 1113: Introduction to C/C++
Programming for Scientists and Engineers
Homework 5
Fall 2018

Due Date: Wednesday, October 31, 2018 before 11:00pm.

Instructions: This is an individual homework assignment. There are two problems worth 20 points each. Solve the problem below by yourself (unlike the labs, where you work collaboratively), and submit the solution as a C++ source code file. Here are a few more important details:

1. Unlike the computer lab exercises, this is not a collaborative assignment.

2. Because all homework assignments are submitted and tested electronically, the following are important:
   - You follow any naming conventions mentioned in the homework instructions.
   - You submit the correct file(s) through Moodle by the due deadline.
   - You follow the example input and output formats exactly given in each problem description.
   - **Regardless of how or where you develop your solutions, your programs compile and execute on cselabs computers running the Linux operating system.**

3. You should test your program on other test cases (that you make up) as well. Making up good test cases is a valuable programming skill, and is part of ensuring your code solution is correct.

Problem A: Commenter (20 points)
Write a C++ program that lets you enter as many sentences as you want (assume less than 100). Keep allowing the user to enter sentences until they type a single line with the word "exit" (no quotes). At this point, you should display everything that they had written, **except** removing an parts that are "comments". Use the normal C++ rules for commenting, where "//" comments out only the rest of the current line and "/*" will continue commenting until a "*/" is reached.

If a comment comments out the full line (no spaces or anything outside of the comment), do not display anything for this line. In other words, fully removed lines should not become "blank" after removing comments, just show the next line that is non-blank. You should also treat "blank" lines in the input the same way, just simply ignore them and do not show them in the output.

For this part, you may assume there is at most one "comment" indication per line. In other words, you will not have a sentence like:

```
I /* like */ apples
```

... you may also assume that every "/*" will have a closing "*/" and that "exit" will not be typed in the middle of a comment. For example, we will **not** test for something like this (no tabs/spaces in input):

```
/* start comment
exit
did I catch you? */
```

Example 1 (user input is underlined):

```
hi there
you are my friend
lets have fun // just kidding, you are a total jerkface
lets /* play
a game
called */ guess
exit
hi there
you are my friend
lets have fun
lets
 guess
```

When you are done, name the source code file <username>_5A.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_5A.cpp. Then submit your program using the HW 5 Problem A submission link in Moodle.

Problem B: Connect Four (20 points)
This part of the homework relates to the Connect-4 code posted on the website. The general purpose of Connect-4 is to be the first person to get 4-in-a-row. This game has historically been played where you slid chips into columns, so the play is always the lowest available row in the column. See this for more details if you are unfamiliar with the rules:

https://en.wikipedia.org/wiki/Connect_Four

The code is complete, except for the "fourInARow()" function. This function is supposed to check and see if there are four consecutive tiles/pieces in the direction(s) indicated. Without this code, players cannot win (but a tie is possible if you fill up the board). Complete this function to correctly detect if there is "four in a row" at the indicated spot in the game.

You should not add any couts in your final submission. You may add temporary couts to help debug, but please ensure these are all removed when you do the final submission.

Example 1 (type in "1 <enter> 2 <enter> 1 <enter> 2 <enter> 1 <enter> 2 <enter> 1 <enter>"):

```
.......
.......
X......
XO.....
XO.....
XO.....
1234567

The mighty Xs reign supreme!
```

Example 2 (type in “2 <enter> 1 <enter> 3 <enter> 2 <enter> 3 <enter> 3 <enter> 4 <enter> 4 <enter> 4 <enter> 4”):

```
.......
.......
...O...
..OX...
.OXO...
OXXX...
1234567

The Os have completed world domination!
```

When you are done, name the source code file <username>_5B.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_5B.cpp. Then submit your program using the HW 5 Problem B submission link in Moodle.

Problem C: Pro-commentator (5 points extra credit)
Expand part A to as many comment markers per line as the person wants. Note: the “//” has a higher precedence than “/*” as can easily be seen in geany (or any other coding program) as shown in example 2.

Example 1 (user input is underlined):

```
what /* the */ heck /* is */ going /* on */ here
this /* makes */ absolutely */ no /* sense */
exit
what  heck  going  here
this  absolutely */ no
```

Example 2 (user input is underlined):

```
override multi line comment // /*
still here /*
not here
/* still /* not /* here
*/ finally!
exit
override multi line comment
still here
 finally!
```

When you are done, name the source code file <username>_5C.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_5C.cpp. Then submit your program using the HW 5 Problem C submission link in Moodle.