|  |  |  |
| --- | --- | --- |
| |  | | --- | | Lab 8 – C++ Classes  CPSC 1021- Fall 19 | |  |

# Due Date: 3 November, 2019 @ Midnight

# Introduction

In today’s lab, we will be implementing a board class to play a game of tic tac toe. We will also continue to practice with I/O manipulation and stringstreams.

# Lab Objectives

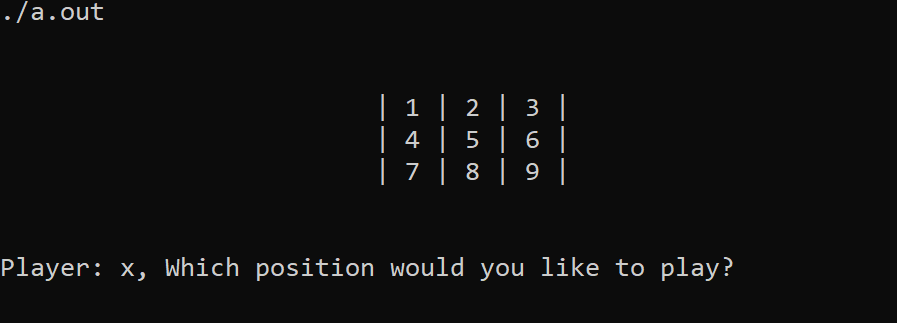
* C++ File I/O + Manipulators
* Writing a Board class
* Object oriented programming
* Using stringstreams to quickly build formatted strings

# Lab Instructions

Create 3 files, main.cpp, makefile, and board.cpp. Your program **does not need command line arguments**, but will take in user input via std::cin. **You may not use printf or scanf** **in your program**.

Sample Output

Before we go any further, here is the expected output for a default gameboard:



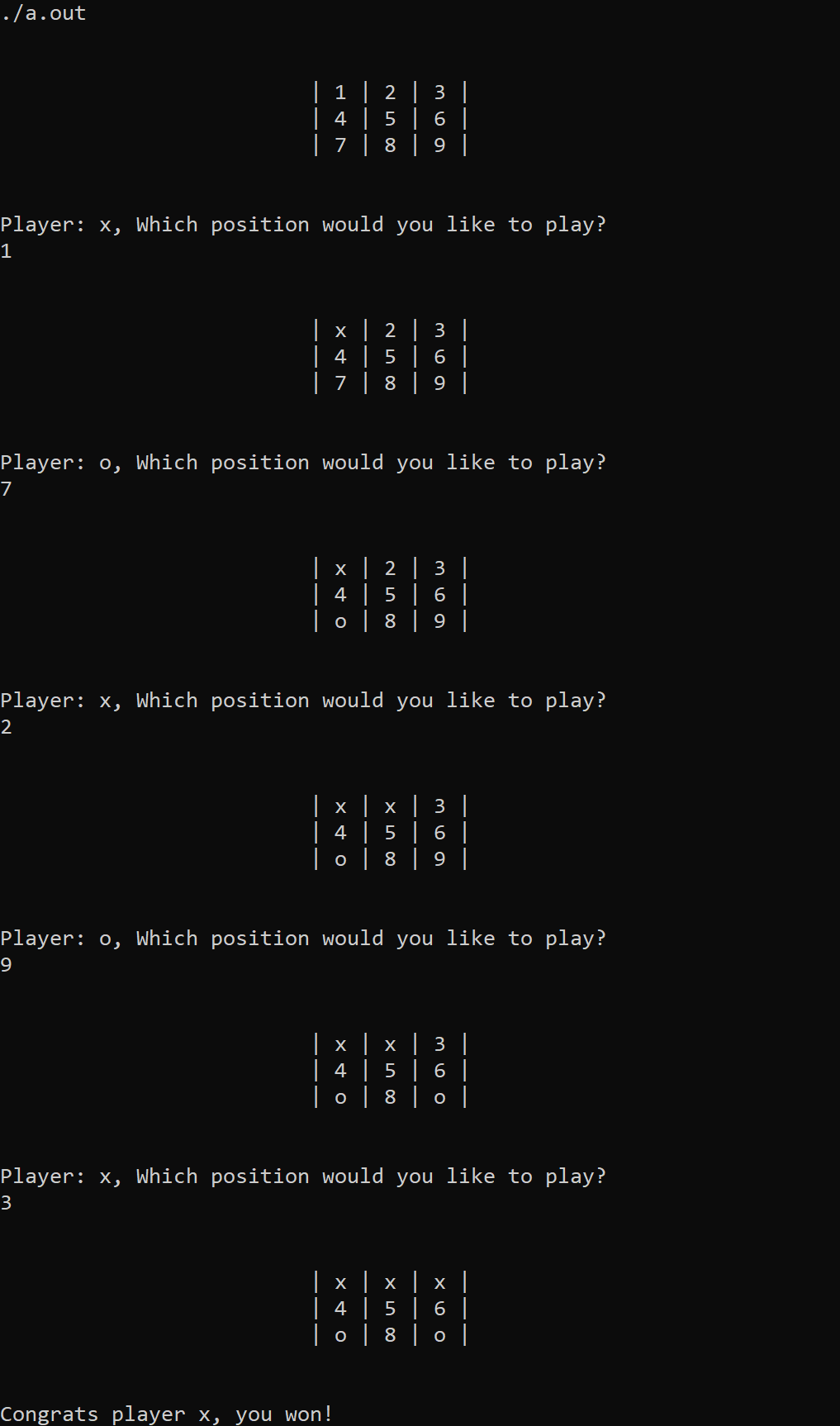
## Board Class

You will be provided with the **board.hpp** file. In C++ your header files can be .h or .hpp. Using the .hpp allows a way to distinguish between C++ and C header files. You must create a board.cpp file and implement all the functions defined in board.hpp. You may create more functions; however, you shouldn’t need to.

The **print** function should use a **stringstream** to build and return a string that you’ll print to the terminal in the main(). See sample output above.

Assumptions you can make:

* Your program does not need to check if there is already a game piece in a given position. Assume we will not test this.
* Your program **does** need to check if we are entering a number between 1 and 9.
* Your program **does need to match my formatting.**

Sample input and output:

## What to turn in

* **main.cpp, board.cpp, board.hpp, makefile**

**SOME HINTS!**

* You can populate the 2D string array using an iterator and the std::to\_string() function.
* Tackle this one function at a time. I recommend starting with the constructor and print() functions.
* You should not need to allocate any memory.
* Draw out every way you can win in tic tac toe and think about how can I program this without brute force checking each win.
* You need to iterate through the vector to get the player. I’d recommend checking if the iterator you use is greater than 1. If it is, reset it to 0.
* When you place a token into the gameboard, do not forget to increment the total number of turns!
* Make sure you test each way a player may win **and for a draw.**

## Compile and Execute

Use g++ to compile your code as follows **and include the C++11 standard!**:

*g++ -std=c++11 -Wall -g main.cpp board.cpp -o game*

Execute the program

### *./game*

|  |
| --- |
| // Sample Header  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  your name  username  Lab 1  Lab Section:  Name of TA  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

FORMATTING:

1. Your program should be well documented
2. Each file should include a header:
3. Your program should consist of proper and consistent indention
4. No lines of code should be more than 80 characters

5 – 10 points will be deducted for each of the above formatting infractions.

## Submission Instructions

* Test your program on the School of Computing server prior to submitting.
* Use the tar utility to tar.gz all source files. **Do not tar an entire directory! When I untar your archive, I should see all the files you included, not a top-level directory! Failure to correctly tar may result in up to a 25-point penalty!**

***EX. tar –czvf yfeaste-lab8.tar.gz \*.cpp \*.hpp makefile***

* Name your tarred file **<username>-lab<#>.tar.gz** (ex. yfeaste-lab8.tar.gz)
* Use handin ([http://handin.cs.clemson.edu)](http://handin.cs.clemson.edu)/) to submit your archive