Midterm Exam

Programming Workshop 2 (CSCI 1061U)

Family name: _____

Given names: _____

Student number: _____

Question	Marks
1	_____ /2
2	_____ /2
3	_____ /4
4	_____ /8
5	_____ /16
Total	_____ /32

Instructions

- All code must be written using a text editor, such as vi, emacs, sublime, notepad++, etc. In other words, please do not use an IDE for this exercise.
- You are only allowed to access the following website(s): http://en.cppreference.com/w/
- Please do **not** access any other resource (past exercises, assignments, labs, books, manuals, etc.) on your laptop, and please do not access any other website.
- Submit via Blackboard.
- This exam has two parts. A written part and a programming part. You need to complete the written part and hand it in before attempting the programming part of the exam.

Written Part

Question 1

Define a 1D character array that can hold upto 10 characters.

Question 2

Change the following piece of code such that it prints the number of commandline arguments passed to the program.

```
#include <iostream>
using namespace std;

int main()
{
  return 0;
}
```

Question 3

Complete the following function to print only even values stored in the array `arr`

```
void print_even(int arr[], int sz)
{
    cout << "Even values are:" << endl;
```

```
}
```

Programming Part

Question 4

Write a program that prints the words stored in a `words.txt` file. Each line of the file stores a single word (with no spaces). We will use the program as follows:

```
$ g++ words.cpp -o words
$ ./words
and
or
this
that
oshawa
$
```

In the above example the `words.txt` file is as follows

```
and
or
this
that
oshawa
```

Starter Code

Use the following starter code for this question

```cpp
// filename: words.cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    fstream f("words.txt");
    if (!f.is_open()) {
        cout << "Cannot open file words.txt" << endl;
    }

    // TO DO

    f.close();
    return 0;
}
```

Question 5

You are asked to develop a c++ program that allows two (human) players to play connect 4, the popular kids game.

> Connect Four (also known as Captain's Mistress, Four Up, Plot Four, Find Four, Four in a Row, Four in a Line and Gravitrips (in Soviet Union)) is a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. Connect Four is a solved game. The first player can always win by playing the right moves.

When we run the program, its output should match the following:

```
$ g++ connect4.cpp -o connect4
$ ./connect4
Board:
Player 1 move: 1
Player 2 move: 2
Player 1 move: 3
Player 2 move: 1
Player 1 move: 2
Player 2 move: 1
Player 1 move: 2
Player 2 move: 1
Player 1 move: 2
Player 2 move: 2
Player 1 move: -1
Exit
Board:
 _ _ _ _ _ _ _
 _ b _ _ _ _ _
 b r _ _ _ _ _
 b r _ _ _ _ _
 b r _ _ _ _ _
 r b r _ _ _ _
```

Usage

The program alternates between player 1 and player 2. Starting with player 1, the program asks for a move. In response, the player 1 enters the column number where he or she wants to drop the colored disks. Valid column numbers are between 1 and 7. If a player enters a -1 at any time, the program exits, and also prints the contents of the board as shown above. A `_` represents an empty slot and `r` and `b` represent filled slots for player 1 and player 2, respectively.

Test files

You can use the following input and output files to test the accuracy of your program.

The input file is:

```
3  4 3 5 3 2 1 -1
```

The output file is:

```
Board:
Player 1 move: Player 2 move: Player 1 move: Player 2 move: Player 1 move: Player 2 move: Player
Board:
 _ _ _ _ _ _ _
 _ _ _ _ _ _ _
 _ _ _ _ _ _ _
 _ _ r _ _ _ _
 _ _ r _ _ _ _
 r b r b b _ _
```

If you run the code as follows:

```
$ ./connect4 < input.txt > foo.txt
```

The `foo.txt` should be exactly the same as `output.txt`. You can check for differences as follows:

```
$ diff foo.txt output.txt
```

If the above command doesn't output anything then your program works.

Starter Code

Use the following starter code.

```
// filename: connect4.cpp
#include <iostream>
using namespace std;

#define NCOLS 7
#define NROWS 6

void print_board(char board[][NCOLS])
{
  cout << "Board:" << endl;

  // TO DO
}

void reset_board(char board[][NCOLS])
{
  cout << "Board:" << endl;

  // TO DO
}

void add(char board[][NCOLS], int ncol, char player)
{
  if (ncol < 0 || ncol >= NCOLS)
  {
```

```
    cout << "Invalid move." << endl;
    return;
  }

  // TO DO

  cout << "Invalid move." << endl;
}

int main()
{
  char board[NROWS][NCOLS];
  char players[] = {'r', 'b'};
  int player=1, col;

  reset_board(board);

  do {
    cout << "Player " << player << " move: ";
    cin >> col;

    if (col >= 0) {

        // TO DO

    }
    else if (col < 0) {
      cout << "Exit" << endl;
      break;
    }
  } while(true);

  print_board(board);
}
```

Submission

Submit Q4 and Q5 via Blackbaord.