**CSCE 2004 – Programming Project 6**
**Final Due Date – Apr 21, 2017 at 11:59pm**

## 1. Problem Statement:

The primary goal of this programming assignment is to give students experience using pre-existing classes. You will be given a Connect4 class that represents the classic 6x7 Connect4 board using a 2D array of characters. The Connect4 class also provides methods to clear the board, let a player make a move, display the board, and finally check to see if one player has won the game. The Connect4 class definition is given below:

```
// Global constants
const int WIN = 4;
const int ROWS = 6;
const int COLS = 7;

// Class definition
class Connect4
{
   public:
      // Constructor and destructor
      Connect4();
      ~Connect4();

      // Public methods
      void ClearBoard();
      bool MakeMove(int col, char player);
      bool CheckWin(char player);
      void PrintBoard();

   private:
     // Private variables
     char board[ROWS][COLS];
     int count[COLS];
};
```

Your task is to write an interactive program that will allow the user to play the Connect4 game against the computer. To do this, you must implement a "game loop" that lets the user and the computer take alternate turns placing their pieces on the board until one player or the other wins the game (or there are no spaces left to play in). To keep things simple, your computer player (the game AI) should choose a column to play in at random. Your program should display the board after every move, so the player can plan their strategy for winning the game.

**2. Design:**

For this assignment, the major design decisions were made when the Connect4 class was created. Notice that there are global constants that specify how many rows and columns there are on the board, and a private 2D array called "board" that contains characters representing the player's pieces on the board. There is also a private 1D array called "count" that keeps track of how many pieces have been placed in each column (which is helpful when placing pieces on the board).

The two methods "ClearBoard" and "PrintBoard" are very simple, and do as their names suggest. The method "MakeMove" checks to see if the specified column has room, and if so, places the player's piece at the top of the column. If not, the method returns false. The method "CheckWin" checks all of the rows, columns, and diagonals to see if the specified player has four pieces in a row anywhere. If so it returns true, if not it returns false.

Your primary design decision is to figure out what the "game loop" will look like from the user's perspective. In particular, how the user and computer will take turns playing pieces, what the user prompts will be, and what the program will output.

**3. Implementation:**

You are starting this programming project with the definition and implementation of the Connect4 class. Your first task should be to write a simple main program that creates a Connect4 object, and performs a collection of method calls to learn how to use this class.

Once you are confident in how the methods work, you can implement the "game loop" that prompts the user for their move, prints the board, makes a random move, prints the board, and checks to see if either player has won.

As always, it is very important to make changes incrementally one feature at a time, writing comments, adding code, compiling, and debugging. This way, you always have a program that "does something" even if it is not complete.

**4. Testing:**

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

**5. Documentation:**

When you have completed your C++ program, write a short report using the "Programming Project Report Template" describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this project report in a separate document to be submitted electronically.

**6. Project Submission:**

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and your C++ program (a single cpp or txt file). Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

10% off if less than 1 day late,
20% off if less than 2 days late,
30% off if less than 3 days late,
no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

**7. Academic Honesty Statement:**

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.