PROGRAMMING ASSIGNMENT #2 - REVIEW

```
using System;

namespace LWTech.ChipAnderson.ConnectFour
{
   class Program
   {
      static Random rng = new Random();

      enum Square
      {
         Empty, Red, Black
      }

      const int GridSizeX = 7;
      const int GridSizeY = 6;
      const int WinningRunSize = 4;

      static void Main(string[] args)
      {

         Console.WriteLine("Connect Four \t\t\t\t Chip Anderson");
         Console.WriteLine("=============================================================");
         Console.WriteLine();

         Square[,] theGrid = new Square[GridSizeX, GridSizeY];       // Lower left corner is 0,0.  Upper right corner i
s 6,5

         // Initialize the grid
         InitializeGrid(theGrid);

         // Initialize the game statistics
         int round = 0;
         int column = 0;
         int numCheckers = 0;
         bool isFinished = false;
         bool humanWon = false;
```

```
bool computerWon = false;

// While there isn't a winner
while (!isFinished)
{
   round++;
   Console.WriteLine("\nRound #: " + round);
   Console.WriteLine("It's your turn!  Here's the current grid:");

   // Display the grid
   PrintGrid(theGrid);

   // Human player takes a turn
   column = ChooseColumnHuman(theGrid);
   if (DropChecker(theGrid, column, Square.Red))
      numCheckers++;

   // Display the grid
   PrintGrid(theGrid);

   // Did the Human win?
   if (WinnerFound(theGrid))
   {
      humanWon = true;
      isFinished = true;
      break;
   }

   Console.WriteLine("\nIt's the computer's turn.");

   // Computer player takes a turn
   column = ChooseColumnComputer(theGrid);
   Console.WriteLine("The computer drops a checker in column #" + (column+1));
   if (DropChecker(theGrid, column, Square.Black))
      numCheckers++;

   // Display the grid
   PrintGrid(theGrid);

   // Did the Computer win?
   if (WinnerFound(theGrid))
   {
```

```
            computerWon = true;
            isFinished = true;
            break;
        }

        // Is the grid full?
        if (numCheckers == GridSizeX * GridSizeY)
            isFinished = true;
    }

    // Display the results
    if (humanWon)
    {
        Console.WriteLine("WOOT! Good job! Humans rock!");
    }
    else if (computerWon)
    {
        Console.WriteLine("Better luck next time meatbag!");
    }
    else
    {
        Console.WriteLine("It's a TIE!");
    }
}

private static void InitializeGrid(Square[,] grid)
{
    for (int j = 0; j < GridSizeY; j++)
    {
        for (int i = 0; i < GridSizeX; i++)
        {
            grid[i, j] = Square.Empty;
        }
    }
}

private static void PrintGrid(Square[,] grid)
{
    Console.WriteLine("+---+---+---+---+---+---+---+---+");
    for (int j = GridSizeY; j > 0; j--)
    {
        for (int i = 0; i < GridSizeX; i++)
```

```
            {
                char c = " xo"[(int)grid[i, j-1]];
                Console.Write("| " + c + " ");
            }
            Console.WriteLine("|\n+---+---+---+---+---+---+---+");
        }
        Console.WriteLine();
    }

    private static bool DropChecker(Square[,] grid, int column, Square checker)
    {
        int j = 0;
        while (grid[column, j] != Square.Empty)
            j++;
        if (j < GridSizeY)
        {
            grid[column, j] = checker;
            return true;
        }
        else
        {
            Console.WriteLine("You bonehead. That column is full!");
            return false;
        }
    }

    private static int ChooseColumnHuman(Square[,] grid)
    {
        int column = 0;
        do
        {
            // column = rng.Next(GridSizeX);          // When testing, uncomment this line and comment out everythi
ng else in this loop.
            Console.WriteLine("Please enter a column # for your checker (1-" + GridSizeX + "): ");
            string s = Console.ReadLine();
            while (!int.TryParse(s, out column) || column < 1 || column > GridSizeX)
            {
                Console.WriteLine("I didn't understand your entry.  Please enter a number between 1 and " + GridSize
X + ":");
                s = Console.ReadLine();
            }
            column--;
```

```
    } while (grid[column, GridSizeY-1] != Square.Empty);

    return column;
}


private static int ChooseColumnComputer(Square[,] grid)
{
    int column = 0;
    do
    {
        column = rng.Next(GridSizeX);
    } while (grid[column, GridSizeY-1] != Square.Empty);

    return column;
}


private static bool WinnerFound(Square[,] grid)
{
    int ajacentCheckers;
    Square curChecker;

    // Scan the columns from bottom to top for 4 vertical checkers
    for (int col = 0; col < GridSizeX; col++)
    {
        curChecker = grid[col, 0];
        ajacentCheckers = 1;
        for (int row = 1; row < GridSizeY; row++)
        {
            if (grid[col, row] == Square.Empty)
                break;              // There cannot be other checkers above an empty square in a column

            if (grid[col, row] == curChecker)
            {
                ajacentCheckers++;
                if (ajacentCheckers == WinningRunSize)
                    return true;
            }
            else
            {
                curChecker = grid[col, row];
```

```
                    ajacentCheckers = 1;
                }
            }
        }

        // Scan the rows for 4 horizontal checkers
        for (int row = 0; row < GridSizeY; row++)
        {
            curChecker = grid[0, row];
            ajacentCheckers = 1;
            for (int col = 1; col < GridSizeX; col++)
            {
                if (grid[col, row] == curChecker)
                {
                    if (curChecker != Square.Empty)
                    {
                        ajacentCheckers++;
                        if (ajacentCheckers == WinningRunSize)
                            return true;
                    }
                }
                else
                {
                    curChecker = grid[col, row];
                    ajacentCheckers = 1;
                }
            }
        }

        return false;
    }
  }
}
```