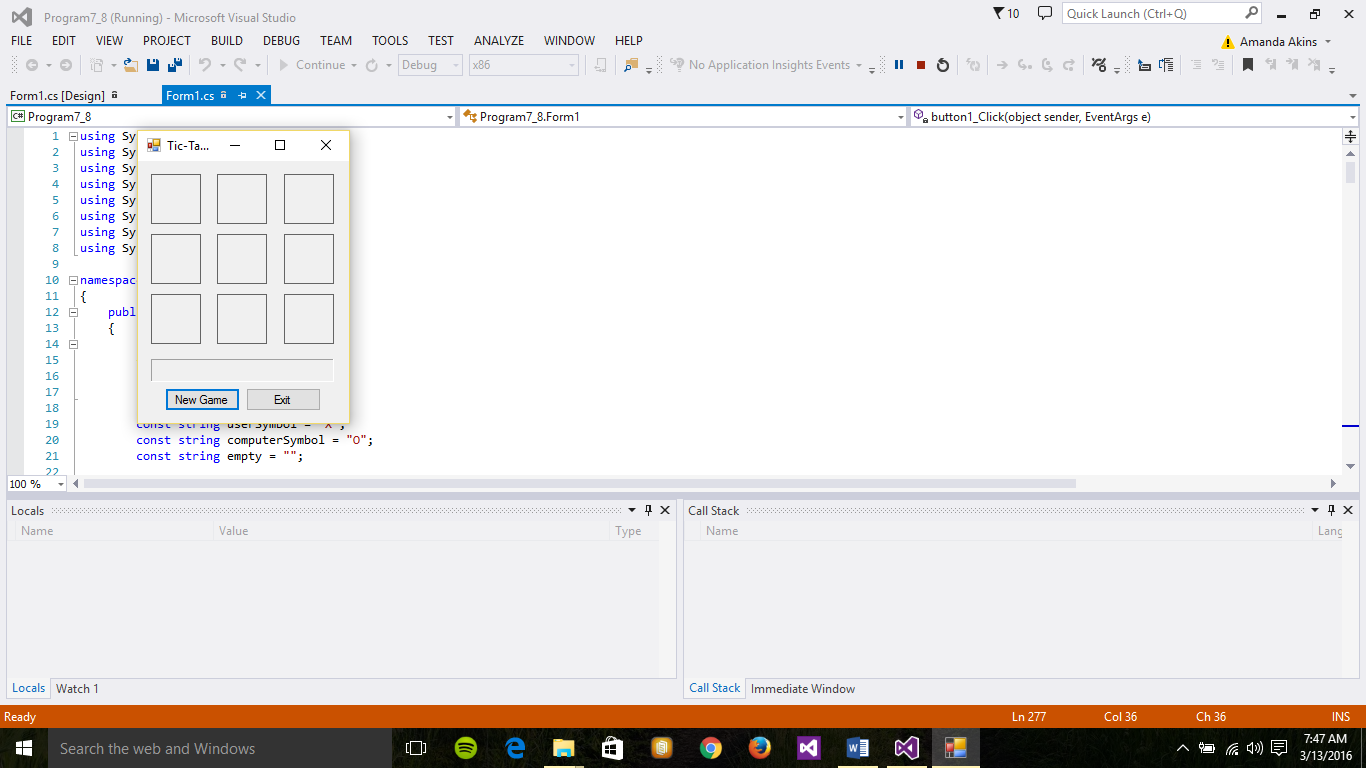
**Lab 12 – CS 133N NAME: Amanda Akins**

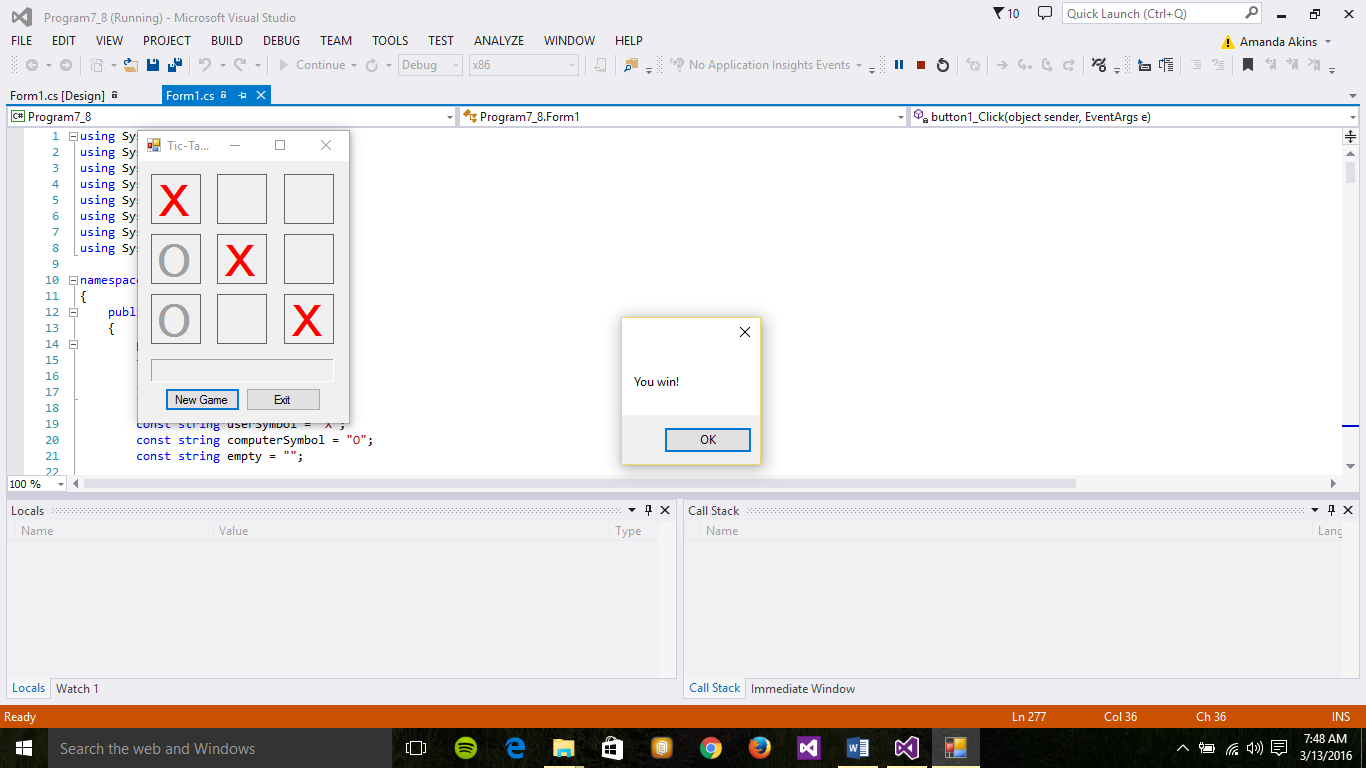
The objective of this lab is to give you additional practice in writing and calling methods in C#.  This information is in chapter 6 of your textbook.

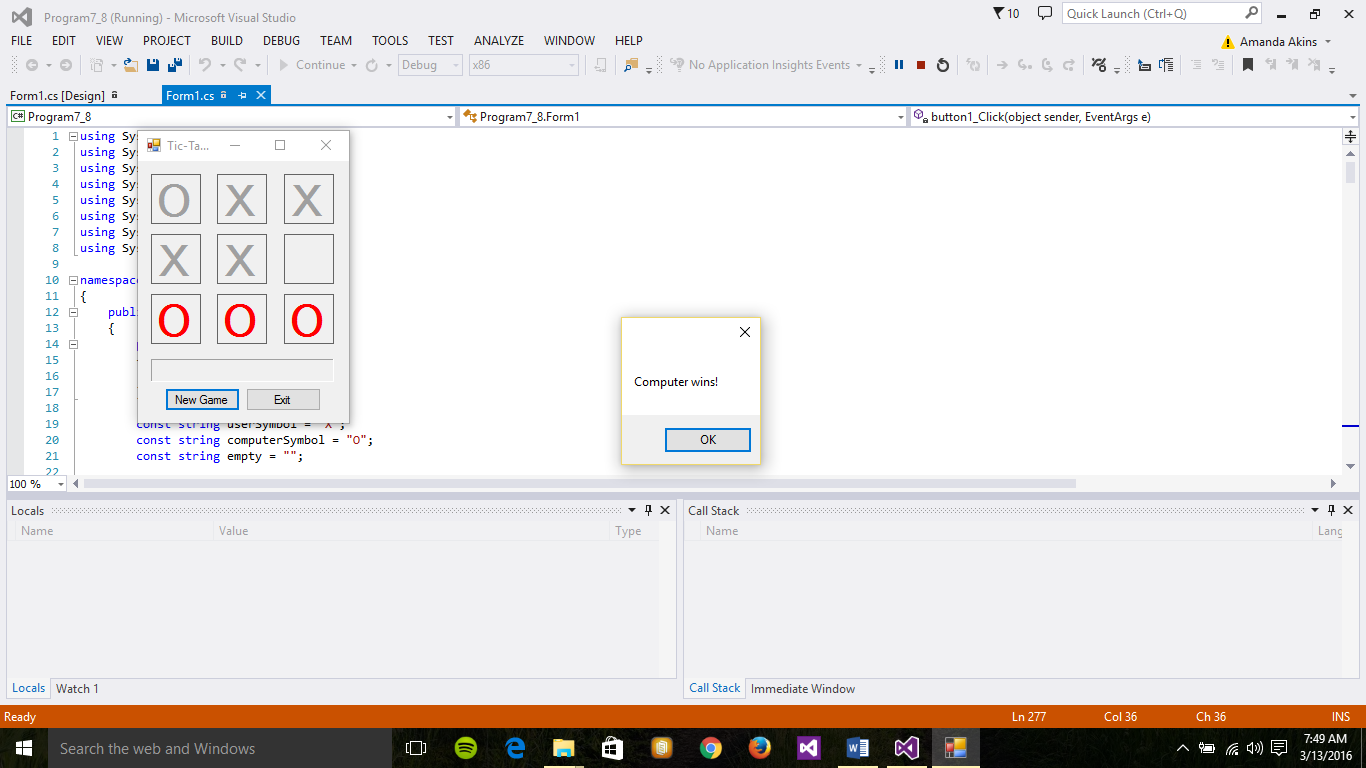
Complete a version of the Tic Tac Toe game beginning with the starting files.  A user interface as well as several methods and the event handlers have been created for you.  I've also given you the heading for all of the methods that I would write if I were solving this problem.  While you don't have to use the code I've given you, your solution MUST use at least 10 methods.

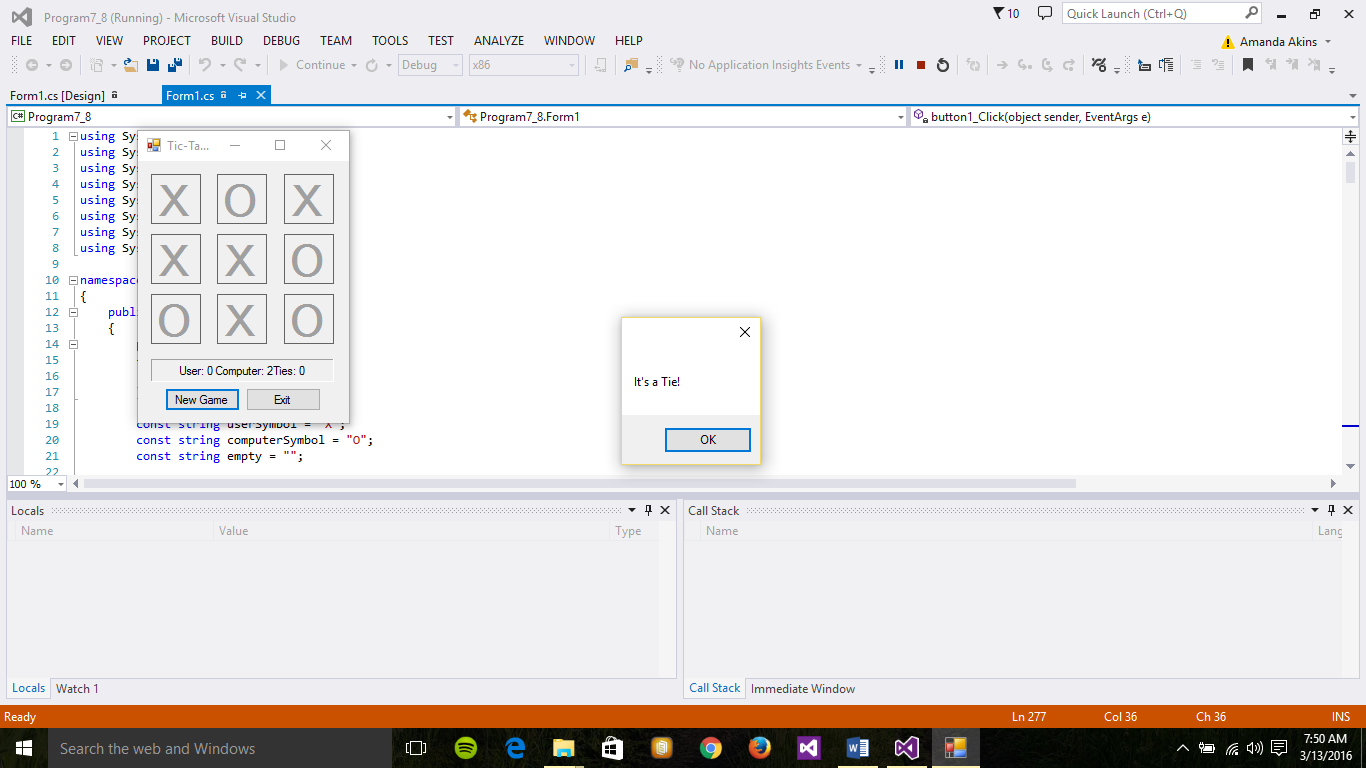
What you should upload to Moodle:

* A screen shots of Tic Tac Toe that demonstrate the functionality of your program:









* The source code for the methods and event handlers that you used in your program (even those that were provided to you:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Program7\_8

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

const string userSymbol = "X";

const string computerSymbol = "O";

const string empty = "";

int userWins = 0; // To count number of wins for user.

int computerWins = 0; // Used to count number of wins for computer.

int countTies = 0; // Used to count the number times there is a tie.

private Label GetSquare(int row, int column) // Takes row and column it receives.

{

int labelNumber = row \* 3 + column + 1;

return (Label)(this.Controls["label" + labelNumber.ToString()]); // "Controls" object is an array of all controls in program. "Label" is saying take "this" control and make it a label.

}

private void GetRowAndColumn(Label l, out int row, out int column)

{

int position = int.Parse(l.Name.Substring(5)); // There is a "Name" property of every label; "Substring" grabs the sixth character; assigns to "position."

row = (position - 1) / 3; // Row 1 column 1 using integer division.

column = (position - 1) % 3; // Remainder == 1.

}

// Set the Text, Enabled and ForeColor properties of

private void ResetBoard()

{

for (int r = 0; r < 3; r++)

{

for (int c = 0; c < 3; c++)

{

Label l = GetSquare(r, c);

l.Text = "";

l.Enabled = true;

l.ForeColor = Color.Black;

}

}

}

// Hightlight the winning column by changing the font

private void HighlightColumn(int col)

{

for (int row = 0; row < 3; row++)

{

Label square = GetSquare(row, col);

square.Enabled = true;

square.ForeColor = Color.Red;

}

}

// Hightlight the left to right diagonal by changing

private void HighlightDiagonal1()

{

int row = 0;

int col = 0;

while (row < 3 && col < 3)

{

Label square = GetSquare(row, col);

square.Enabled = true;

square.ForeColor = Color.Red;

row++;

col++;

}

}

// Hightlight the right to left diagonal by changing

private void HighlightDiagonal2()

{

int row = 0;

int col = 2;

while (row < 3 && col >= 0)

{

Label square = GetSquare(row, col);

square.Enabled = true;

square.ForeColor = Color.Red;

row++;

col--;

}

}

private void HighlightRow(int row)

{

for (int col = 0; col < 3; col++)

{

Label square = GetSquare(row, col);

square.Enabled = true;

square.ForeColor = Color.Red;

}

}

private void MakeComputerMove()

{

// do

// pick a random number between 0 and 2 for row

// and another random number between 0 and 2 for column

// Repeat this process if that square has a value in it

// FYI use GetSquare method to "find" the label at row, column

// disable the square

// set the text to the computer's symbol

// if computer won (IsWinner)

// display a message box

// else if it's a tie

// display a message box

Random randomNumberGenerater = new Random();

int row;

int col;

Label selectedLabel;

do

{

row = randomNumberGenerater.Next(0, 3); // Randomly assign 0, 1 or 2 to the row.

col = randomNumberGenerater.Next(0, 3); // Randomly assign 0, 1 or 2 to the column.

selectedLabel = GetSquare(row, col); // Receives the label of the square selector.

}

while (selectedLabel.Text != "");

selectedLabel.Text = computerSymbol.ToString(); // Set the text to the computer's symbol.

selectedLabel.Enabled = false; // disable the square so it connot be selected.

if (IsWinner())

{

MessageBox.Show("Computer wins!");

computerWins++;

label10.Text = "User: " + userWins + " Computer: " + computerWins + "Ties: " + countTies;

}

else if (IsFull())

{

MessageBox.Show("It's a Tie!");

countTies++;

label10.Text = "User: " + userWins + " Computer: " + computerWins + "Ties: " + countTies;

}

}

// Returns true if the provided row has all Xs or Os, otherwise returns false.

private bool IsRowWinner(int row)

{

// if square(row, 0) = square(row, 1) = square(row, 2) AND square(row, 0) is not empty

// highlight the row

// return true

// else

// return false

if (GetSquare(row, 0).Text == GetSquare(row, 1).Text && GetSquare(row, 0).Text == GetSquare(row, 2).Text && GetSquare(row, 0).Text != "")

{

HighlightRow(row);

return true;

}

else

{

return false;

}

}

// Returns true if any of the 3 rows is found to be a winner.

private bool IsAnyRowWinner()

{

bool winner = false; // Set to true if there is a winner based on a row.

for (int row = 0; row < 3; row++)

{

if (IsRowWinner(row))

winner = true;

}

return winner;

}

// Returns true if the provided column has all Xs or Os, otherwise returns false.

private bool IsColumnWinner(int col)

{

if (GetSquare(0, col).Text == GetSquare(1, col).Text && GetSquare(0, col).Text == GetSquare(2, col).Text && GetSquare(0, col).Text != "")

{

HighlightColumn(col);

return true;

}

else

{

return false;

}

}

private bool IsAnyColumnWinner()

{

bool winner = false; // Set to true if there is a winner based on a row.

for (int col = 0; col < 3; col++)

{

if (IsColumnWinner(col))

winner = true;

}

return winner;

}

// Returns true if either of the two diagonals have all Xs or Os, otherwise returns false.

private bool IsAnyDiagonalWinner()

{

if (GetSquare(0, 0).Text == GetSquare(1, 1).Text && GetSquare(0, 0).Text == GetSquare(2, 2).Text && GetSquare(0, 0).Text != "")

{

HighlightDiagonal1();

return true;

}

else if (GetSquare(0, 2).Text == GetSquare(1, 1).Text && GetSquare(0, 2).Text == GetSquare(2, 0).Text && GetSquare(0, 2).Text != "")

{

HighlightDiagonal2();

return true;

}

else

{

return false;

}

}

// Returns true if any row, column or diagonal has all Xs or Os, otherwise returns false.

private bool IsWinner()

{

if (IsAnyRowWinner() || IsAnyColumnWinner() || IsAnyDiagonalWinner())

{

return true;

}

else

{

return false;

}

}

// Returns true if all 9 squares have an X or an O. If this occurs neither player won.

private bool IsFull()

{

bool full = true; // Set to false when encounter an empty string is a cell.

for (int row = 0; row < 3; row++)

{

for (int col = 0; col < 3; col++)

{

Label lab = GetSquare(row, col);

if (lab.Text == "")

{

full = false;

}

}

}

return full;

}

private void label\_DoubleClick(object sender, EventArgs e)

{

Label clickedLabel = (Label)sender; // Creates a label object.

if (clickedLabel.Text == "") // If the text property of the clickedLabel object is empty, do the following if statement, if no x or o in label can't click.

{

int row, column;

GetRowAndColumn(clickedLabel, out row, out column); // Passing a label and two integers. Get row and column of where user clicked, "out" we get back.

clickedLabel.Text = userSymbol.ToString(); // Setting "X" to the text property of label user clicked on.

clickedLabel.Enabled = false; // Sets "Enabled" propery to false to prevent label from being clicked it again.

if (IsWinner()) // Returns true.

MessageBox.Show("You win!");

else if (IsFull()) // Returns false.

MessageBox.Show("It's a Tie!");

else

MakeComputerMove();

}

}

private void button1\_Click(object sender, EventArgs e)

{

// Clear out gameboard.

ResetBoard();

}

private void button2\_Click(object sender, EventArgs e)

{

// Close the game.

this.Close();

}

}

}