**University of Leeds** **School of Computing**
Procedural Programming COMP1711/XJCO1711
Semester 1, 2020-2021

# Coursework 2

80 Marks (40% of the total module mark)

**To be submitted before: 23:00 (UK time) on 13 November 2020**
**Late penalties: 5% will be deducted from your overall mark for every late day**

## Skills Tested

This coursework will test your ability to write C code that correctly uses loops, arrays, and functions.

## The Brief

You will write a program to implement an extended version of the famous Tic-tac-toe game. The program can support two players competing against each other. The program will display the game board, record moves, detect winners, and playback the game when it is over.

## The Details

### The Problem

Tic-tac-toe, also known as noughts and crosses or Xs and Os, is a simple paper and pencil game for two players. A grid (matrix) of *nxn* cells is first drawn. Each player chooses one of two possible symbols (usually X or O). The two players then take turns in drawing the symbols within the cells of the grid. The player who first succeeds in forming a contiguous horizontal, vertical, or diagonal line of symbols wins the game, as shown in the figure below.

The most famous version of the game uses a 3x3 grid, and the player who first forms a line of 3 consecutive symbols wins the game. However, an extended version of the game can be played with any size of grid and any number of consecutive symbols required for a win. For example, the game can be played with a grid of 10x10 cells, with the winner being the one who makes a line of 6 consecutive symbols in any direction.

You will write a program to help players play this game. The program will display the empty grid, organize turns, and detect a win as soon as it occurs. It will also record all moves during the game then

permits the players to watch a playback of the game at the end. The program should support any grid size from 3x3 to 10x10, and any length for the contiguous line of symbols required for a win.

## Program operation

When the program starts, it will prompt the user to enter the size of the grid (between 3 and 10). It will also prompt the user to enter the number of consecutive symbols a player must make in order to win the game (this number cannot be less than 3 or greater than the grid size). The program then displays the empty grid on the screen. Empty cells are shown as dots. To determine cell positions, the program also displays the zero-based row and column numbers (indices) of the cells as shown in the figure below:

```
Enter grid size (3-10):4
Enter winning length (3-4):4

        0 1 2 3

0       . . . .
1       . . . .
2       . . . .
3       . . . .

Player X, Choose Location (row,col):
```

The program then alternately prompts the two players, called Player X and Player O, to place symbols in the grid. The player must enter the coordinates of the cell in which they wish to put their symbol by giving the cell's row and column numbers separated by a comma (named x, y in the figure above) **with the row number (x coordinate) first** , as shown below:

```
Player X, Choose Location (row,col): 1,1

        0 1 2

0       . . .
1       . X .
2       . . .

Player O, Choose Location (row,col): 0,1

        0 1 2

0       . O .
1       . X .
2       . . .

Player X, Choose Location (row,col): 0,0

        0 1 2

0       X O .
1       . X .
2       . . .

Player O, Choose Location (row,col): 2,2
```

The program must check the input and make sure that the coordinates are within range of the grid, and that the intended cell is empty (i.e. a symbol is not being placed on top of an existing one). If the input values are invalid, the program prompts the user to enter the coordinates again, as shown in the figure below.

```
        0 1 2

0       O . .
1       . X .
2       . . .

Player X, Choose Location (row,col): 1,3
Index out of range, please re-enter
Player X, Choose Location (row,col): 0,0
This location is already taken
Player X, Choose Location (row,col):
```

After each 'move' by a player, the program checks if this player has won (i.e. has formed a horizontal, vertical, or diagonal line with the required number of symbols). And, if the player has won, the program declares this player as the winner and stops, as shown below.

```
Player X, Choose Location (row,col): 0,2

        0 1 2

0       O O X
1       . X .
2       X . .

**************************
Player X has won the game
**************************
```

If the grid becomes full and neither player has formed a winning line, the program declares a draw and stops, as shown below:

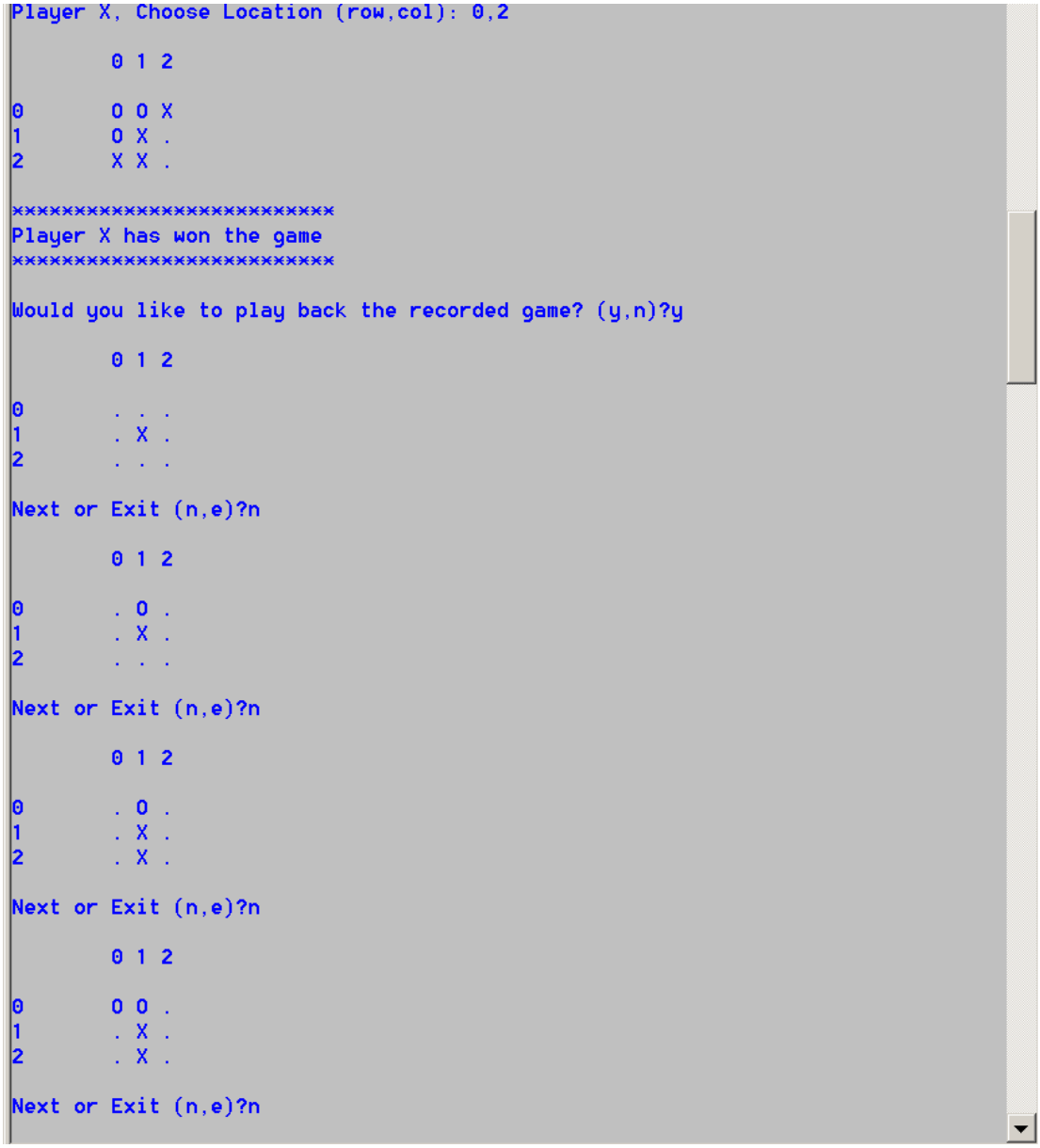
```
Player X, Choose Location (row,col): 2,1

        0 1 2

0       X O X
1       O X X
2       O X O

Game over; there are no winners
```

Once the game is over, the program asks if the user wants to watch a playback of the game, and if the response is y (for yes), the program displays the game's grid after the first move was made, then pauses and asks if the user wants to view the grid after the next move. This continues until all the moves were shown or the user decides to terminate the playback by responding with n, as shown below:




## Implementation Instructions

- Write the program in standard C. If you write your code in any other language, it will NOT be marked, and you will earn a zero mark in this coursework.
- Two files - called tic.h and tic.c - are provided for you to start developing your code. Do not alter anything in tic.h and do not submit it to Gradescope.
- File tic.h contains some function prototypes. You must implement the body of these functions in the tic.c file. You will also need to use these functions in the rest of your code.
- Gradescope will be used to mark this coursework. Detailed instructions on how to prepare and submit your code to Gradescope will be published on Minerva in due course.
- Do not change the name of the code file (the tic.c file) as this will cause your program to fail all Gradescope tests.
- Do not use any scanf or printf statements within the body of the functions that are tested by Gradescope as this will cause the autograder to fail. If you have inserted some scanf or printf statements for debugging purposes, please remove them before submitting your file to Gradescope. It is fine to use printf or scandf in your own functions that are not tested by

Gradesope.

- Make sure that you fill in your name, University number, and the date you started working on the assignment in the space provided at the top of tic.c.
- This is an individual project, and you are not supposed to work in groups or pairs with other students.
- **Please be aware that plagiarism in your code will earn you a zero mark and will have very serious consequences. It is much better to submit your own partially finished work, than to fall into the trap of plagiarism. We use software to detect plagiarism, so if you do work with someone else, or submit someone else's work it WILL be detected.**

## Marking Scheme

All features will be automatically tested by Gradescope. The marking scheme is as follows:

| | |
|---|---|
| Function `newGame()` | (8 Marks) |
| Function `makeMove()` | (8 Marks) |
| Function `boardIsFull()` | (4 marks) |
| Function `checkHorizontal()` | (8 marks) |
| Function `checkVertical()` | (4 marks) |
| Function `checkDiagonals()` | (8 marks) |
| Function `checkAntiDiagonals()` | (4 marks) |
| Function `playerHasWon()` | (6 marks) |
| Function `effPlayerHasWon ()` | (6 marks) |
| Function `showGrid()` | (8 Marks) |
| The user interface works correctly | (8 marks) |
| The program can play back recorded games | (8 marks) |
| Total | **[80 marks]** |