40337540 – Algorithms & Data Structures Report

**Introduction:** The given task was to implement a game of Tic-Tac-Toe that is playable on the Console, programmed in C. We had to implement several features and focus on the data structures and algorithms used to implement the core game and its extended features. Some of the required features for the core game were; Drawing a game board to the screen, The ability for input from the two players, Having the unique pieces on the screen (The X’s and the O’s) and each position on the board having its own unique identifier so players do not get confused on which position they’re going to choose. The extended features for this task were recording the history of previous played games so they can be watched over, The ability to Undo and Redo moves and finally the ability to choose a custom sized board (diverging from the traditional 3x3 playing board).

**Design:** At the head of my code I made 3 macros to create the constants used for the board size and the max length for the input by the user. I mainly used arrays for designing my game, as it allowed me to quickly and easily store and retrieve all the necessary data. When designing the standard board, I added each cell into a 3d array, dividing it into rows and columns so each cell has its own unique identifier, I then made an array combining all the possible winning combinations for the players. For algorithms I had my program check which player is currently playing (player 1 or player 2) and check which cell they have attempted to place their move and if the cell is already occupied by an X or O. If the cell is occupied it loops until the player enters a valid unoccupied cell and adds it to the ‘currentgame’ array. Then the game updates the board to reflect the most recent move and lets the other player have their go.

The other unique algorithms I made were the ability to undo and redo moves, for the undo move method it gets the currentgame array, getting the highest index and making it null. It then updates the board to reflect this most recent change. The redo method works very similar to the undo method, it simply gets which move was recently undid and reverts the board to its previous state, adding it back onto the board.

Since I used arrays to store all the moves and which player made the moves it has given me the ability to implement the replay functionality quite easily, my program simply gets the arrays from the previous games and shows a step by step of each move the players did until it runs out of games to replay. An added benefit playing from the undo/redo methods is that it doesn’t show the altered moves (as they are overwritten during the game process) and only shows the final move each player did for their respective turn.

**Enhancements:**  My game in its current form is the classic Tic-Tac-Toe formula everyone has played at some point during their lives, but on the Console and in C instead of lined paper. A simple improvement to the game would be the ability to make a custom sized board, expanding on the default 3x3 board, for example having pre-set 4x4 5x5 6x6 ect or choosing your own size board eg 5x12. To expand on the customisation aspect custom win conditions would be an interesting feature to implement, so the players may need 4 in a row, or 5, 6 ,7 depending on the board size as well as making minimum win conditions. So users cant make it so they only need 2 in a line/diagonal if the board is a 10x10 and the minimum requirement is a 4 in line/diagonal. Another unique feature that I would implement if I had more time would be giving the user the ability to enter a custom player icon, so instead of X’s and O’s, you could have anything from ‘@’ to using a ‘$’.

A final improvement I feel would be nice for the users customisation would be the implementation of a leaderboard that saves the amount of wins and losses for a specific user. So player 1 could make an account and store their wins to see which player truly is the Tic-Tac-Toe champion and compete for the top spot, everyone loves a bit of competition and would try to play optimally since their stats are on the line.

**Critical Evaluation:** While I do think my application runs well, accomplishing each task accurately and without errors, bugs or crashing. The players are unable to enter invalid values when entering their move, you can play multiple games in one sitting without having any problems and you are able to undo and redo moves with the simple press of a button (u and r respectively). I used limited data structures in the making of my application. I mainly used arrays, while this made certain aspects of the app easy to program and work with it has limited my ability to make the playboard variable sizes since an array can only be a static size, therefore without creating a larger array the game would try to add a move to an already full array and inevitably crash the game.

Another feature that doesn’t work perfectly is the replay function, it does work and replays the games that have been played in the current sitting it quickly cycles through each move made by the players too fast to watch properly. This occurs because the replay feature is simply playing the game itself using the moves stored in the array for the game. The problem occurs because after each move the screen clears itself and redraws the board and showing the move it has just done. While this makes the UI a lot cleaner when playing the game, it makes watching replays suboptimal and may ruin the experience for people wanting to watch the replays.

**Personal Evaluation:**  While making this application I learnt how to make and use Macros in C as well as work using pointers for variables when passing them into methods. Also learning %d and %c in printf statements help to make it more readable and made it easier to pass variables into any print statement or read the input from the command line.

I learnt how to add, remove, modify data in arrays since I used arrays to store all the moves the players make I worked using arrays a lot and have a further understanding on how they work and how to use them a lot more effectively.

Some of the problems I faced was when it came to working with nested if statements and else ifs, this was only a problem due to me messing up some of the braces and indentation of my code. This was an easy fix as it just meant I had to be more careful when typing the code out.

Another problem I was facing was the UI being cluttered since my game redraws the board after every move adding the most recent move. I had to figure out how to clear the screen. Although I found online how to easily clear the screen it does only work on Windows which limits it utility.

Overall, I feel I performed very well with an efficient and effective solution. I completed the base game and most of the extended tasks with great success (excluding the replays due to how the game refreshes itself).Besides that, I feel the completed application has a nice clean UI, functions how Tic-Tac-Toe should and doesn’t have any glaring bugs or errors.