Data structures and Algorithms Coursework

# Introduction

The task is to create a text-based Tic-Tac-Toe game by implementing several of data structures and algorithms in the C programming language. The game must have a game board, players, pieces and positions. The game must record the history of play and enable earlier games to automatically replayed from this record. The game must support undo where once a move made, a player can undo the move to the game state before where players can undo moves to initial game state. The game should also have a redo feature that allows redoing moves which been undone. The features that the Tic Tac Toe game has a register and log in where users can register an account by setting a username and password. The password is encrypted then the encrypted and username is written to a file. The login feature is where the user can log in into the game as long the username and password equal to username and password in a particular line of a file. The other features are a multiplayer game where two humans can play with each other by entering the square number that a user wants to position their mark on until a player wins or there is a draw. The game moves are then replayed from the initial state to end state in a way where each position placed every time a user presses a key until finished state. A player can then decide to play again or go back to the menu. There are also two other types of games where a player can play against a computer. One game is where the computer makes a random move after a player makes a move until the game completed. The other version is where a minimax implementation is used to allow a computer to observe all possible moves and chooses the best move every time the computer makes a move. There are two replays search system. One is where a user can enter a game id key and certain moves from games that have played replayed subsequently. Another is where a user can enter a game id key and certain moves from a game from among the games that specific user have played in the past.

# Design

### Tic TAC Toe Game – Multiplayer Design

The tic tac toe game is played on a 3x3 game board by two players, who take turns where the first player moves with a cross and second move with a circle. The player who has formed a horizontal, vertical or diagonal sequence of three marks wins. When the game board is full, and no one wins, it is a draw. The game board implemented by a 1D array of chars where each char represent a square of the board which initialised with values ‘1’, ‘2’, ‘3’, ‘4’, ‘5’,’6’,’7’, ‘8’, ‘9’. Depending on what square number, that a player has positioned a piece on the game board, the string would be replaced with a certain piece, either a nought(‘O’) or cross(‘X’), depending on what player has positioned a move. The Game board implemented as a 1D array instead of a 2D array as 1D array stores data in a list whereas 2D array stores data in a row-column format, Therefore, it is easier to implement and faster loop through the game board and change value of game board square when a player makes a move compare to being implemented by a 2D array. Game board implemented by using an Array instead of a linked list because Array supports Random Access, that indicate elements can be accessed directly using their index. For example, when a player decides to position a cross piece to square 7, square[7] can easily change from ‘7’ to ‘X’ whereas Linked list supports Sequential access, that indicates accessing any square of the game board, we have to traverse the complete linked list to that square sequentially. Therefore, to access the nth element of a linked list, time complexity is O(n) whereas accessing elements within the array is fast with a constant time complexity of O(1). Game board implemented by an array instead of a stack because, in an array, the objects are arranged in a way so it can be accessed at any time randomly whereas, in a stack, the objects arranged in a way where it is only inserted or deleted from one end Insertion and deletion take place within any position. Therefore, cross or nought inserted to the Array implemented game board compare to a Stack implemented game board which would make the game more efficient.

Four stacks are used to implement the undo and redo function. A move is undone every time user types in ‘10’ and a player can redo a move by typing ’11’. The stacks made are a stack that contains all of the choices which are called ‘undoChoices’, stack that contains all of the marks which are called ‘undoMarks’, stack that contains all of choices that been undone which is called ‘redoChoices’ and stack that contains all of the marks that been undone which is called ‘redoMarks’. Every time a player makes a move, choices, in the ‘redoChoices’ destroyed, marks in the ‘redoMarks’ destroyed, choice pushed into the ‘undoChoices’ and a mark pushed into the ‘undoMarks’. When a player (either player 1 or player 2) undo a move, the mark popped from ‘undoMarks’ and pushed into the ‘redoMarks’, the choice is popped from ‘undoChoices’ and pushed into the ‘redoChoices’. The game then set to the previous game state and is the next player turn to make a decision. Players can keep undoing moves from current state till initial game state where the game board is empty. When a user redoes a move, the un-done move will be re-done where players can redo until all undone moves are on the game board. This operation happens by popping a mark from ‘redoMoves’ and pushing into ‘undoStack’, popping the choice from ‘redoChoices’ and push into the ‘redoChoices’. When a player makes a move, the players can no longer redo moves that they previously have undone. Stacks used Instead of arrays to implement these features because a stack is a linear data structure shown by a sequential collection of elements in a fixed order where an array is a collection of related data values called elements each identified by an index array. Also, stack elements can be added or removed in a LIFO order meaning last one in is first to be accessed and first one in can be accessed last whereas an in an array, it is a random access operation, and everything gets down to start of the array. Insertion and deletion take place in any position. Therefore, faster to undo and redo moves by using a stack instead of an array. Stack used instead of a queue to implement the undo and redo moves as it is simpler to implement both features using a stack compare to a Queue. Also, in a stack, the same end is used to insert and delete elements whereas in a Queue, on end is used for insertion such as rear end and another end is used for deletion of elements such as front end which mean stack makes it faster for players to undo or redo moves compare to a Queue. Dynamic Array implantation of the stack is used instead of a singly linked list implementation of the stack because even though both have average O(1) time when a new move added to the linked list, new allocation needed that can be high contrast to other operations. Arrays do not suffer from this problem.

The undoMoves stack array used to replay the game after a player wins the game or there is a draw. The feature works by looping through the array from the first index to the top value of the stack array. For each iteration, a mark into a square and the user is asked to press any key to show the next move that repeats until the last state of the game. The user then asked if he/she wants to play an again by typing in ‘yes’ or any key to not play again. These features are present in the Two Players game, play with the computer(hard) and the player with computer (easy) game. Stack array instead of a Queue because it is more accessible to implement and save memory space as an array used for multiple features. I preferred to use an array instead of a linked list for this feature as it takes up less memory compared to a Linked list.

In order to replay moves of a specific previous game, a user must type in the game id of that specific game. The moves of that game replayed from initial game state to final game state which happens by looping through moves of the game from the first index to the 9th index of the moves array. If the game not found, an error message printed on the console stating ‘Game Id not found’. Feature implemented by writing an array of the undoMoves stack array to a file called ‘list.txt’. The function ‘ReadFromFileToLinkedList()’ that has a variable called ‘count’ initialised to 0. The function reads each line from ‘list.txt’ and each line set to a moves array of a node of a singly linked list called ‘Game’ and node-id set to count value.

The count would iterate every time. A linear search algorithm would iteratively look for the user inputted game id in the linked list. When the game id located, the game would subsequently replay all of the moves from initial game state to final game state. When the linear search algorithm iteratively looked through all of the game ids of the linked list and have not found any game id that is equal to the user input value, an error message shown. The linked list used to store games as a feature in an array, elements are stored in a contiguous memory location or sequential manner within memory whereas, in a linked list, new elements can be stored anywhere within memory where the address of memory location allocated to the new element stored within the previous node of linked list. Hence, producing a link between the two nodes that make it easier for the computer to search for a specific game id. In the array, Insertion and deletion operation takes more time as memory location is consecutive and fixed whereas in a linked list, a new element stored at the first free and available memory location with the only single overhead step of storing the address of memory location within the previous node of linked list. Therefore, inserting each game moves from file to linked list than inserting each move from file to an array. Also, the size of the array must be indicated at the time of array declaration whereas the size of a linked list varies where it grows at runtime while more nodes added to it. Therefore, an unlimited amount of games date can add to the linked list without any problems. Even though searching a game from hash map time complexity (O(1) on average) is better than searching a game from a linked list (O(n) on average), implementing a linked list is less complicated than implementing a hash table. Also, an unlimited amount of games inserted to a linked list without needing to expand the linked list which is not the case with a hash table. Also, a hash function would assign each key to a different bucket, but it is possible that two keys would make an indentical hash causing both keys to point to the same bucket that causes collisions which is not the case with a Linked list.

They have therefore linked list used instead of a hash map. Linear search used instead of a binary search to search for the particular game in the linked list as the time complexity searching linked list linearly is same as searching linked list binary as both need to look through each linked list node as each node is linked to the node before. The linked list is searched iteratively instead of recursively as Recursion has high space complexity and slower than an iterative search function. The feature is similar to the feature where a user can search games that he or she played, but the difference is the filename that the moves of each game stored the name of the user’s username.

For this reason, no one can register an account of a username that already exists. This feature allows each user to look at its past moves and to learn how they can improve in later games. Singly linked list used instead of a doubly linked list because singly linked lists have only one pointer whereas doubly linked lists have two field pointers so singly linked lists occupy less memory than doubly linked lists.

For the (Hard)Play With Computer version of the Tic Tac Toe game, I have used a minimax algorithm to allow the computer to consider all possible moves and choose the optimal move every time the computer is making a move on the board while playing against a human. The findBestMove() evaluates all the possible moves using minimax() and then return the best move the maximiser can make. The minimax() algorithm would check whether or not the current move is more optimal than the best move by considering all possible methodologies the game can go and returns the best value for that move, assuming the human also plays optimally. The isMovesLeft() function check whether the game is over and to make sure there are no moves left by returning true or false. Every time is the computer turn; the computer would place the most optimal move on the board. This way is tough for human players to win against the machine. HUamn players can undo and redo a move, and when the game finished, the game moves subsequently replayed.

For the (Easy)Player With Computer version, the computer randomly chooses what moves to make by choosing a range of number from 1 to 9 that is not already chosen by the human player. This way, the computer can quickly choose a move and occupies less memory. This feature allows players who are not experienced in the Tic Tac Toe game to have the opportunity to improve their game abilities.

A user can register an account by pressing key ‘2’. The registered feature allows users to make an account by writing down username and password. The system would check whether or whether not that the username is already existed by reading through the file ‘accounts.txt’ and compare the first string of each line to the entered username. If the username already exists, the user returned to the menu but if that is not the case, the message saying ‘You have successfully, created an account’. The password encrypted by using a hashing function, and then both username and password are written to the ‘accounts.txt’ in a way where the line has space between username and password. The encrypt() method encrypts the password by subtracting hex value from it. The password encrypted for security reasons such as a person hacking into ‘accounts.txt’ to try access a user account.

A user can log in into their account by pressing key ‘1’. The user can then type into their username and password. The program reads each line ‘accounts.txt’ file and compares the entered username and password with a username and decrypted password from the file. When the inputted username and password match username and password from a line of the file, the user is signed in. When the inputted username and password is not equal to any username and password from the file, an error message is then displayed, and the user returned to the previous menu. When the user signed in, the user can have the choice to Quit, play the multiplayer game, play against the computer(Easy), playing against the computer(hard), search replays(All Games) or search replays(User Games). The Quit option will sign user account off and go to the initial menu which happens when user types in ‘O’. The user can completely leave the game by typing in ‘0’ again.

## Enchancement

There would be many features in place if I had more time. One feature would be allowing each user to choose what game board size they want instead of just playing on the 3x3 game board. Before the game starts, the user would be asked what size of the game board they desire to play on. For example, if the user typed in 6. A 6 x 6 game board would be placed contained 6 rows and 6 columns.

Another feature I would like to implement is a Hall of fame feature which contains the number of games that each player have won and lost. The number of games a player have won, the higher the player would be placed in the game. This would be in a table format where the columns are the player names, Games played, games won, games lost and player levels. Everytime a player wins 50 games, the player level would increase by one. I would like to implement this feature because It would encourage players to play the Tic Tac Toe game multiple times as they would compete with other players.

A feature I would like to implement is a player profile which wold also be shown in a table format that shows player name, player level, games won, games lost, message option and compete option. Players can also post photos and update status. A player profile can be searched by going to the search page and tying their name or can be found on the username profiles. This feature would enhance the software as each player would have the opportunity to express themselves uniquely to other players through their profile.

A message and compete feature would be implemented. This feature would allow users to message and compete players around the world. This can possible by publishing this game on a cloud server where players must have internet connection in order to play against or message players around the globe. This would enhance the Tic Tac Toe game as It would allow Tic Tac Toe enthusiasts to meet players around the globe that have similar interest as them and to try beat as much players as possible which is more enjoyable then two players competing using the same PC.

## Critical Evaluation

In my opinion, the Two players game have worked really well because two players can compete each other in a way that they take turn making a move where one player places a cross and another player places a circle on the game board until a player wins or both draw without any cheating taking place.

The undo and redo feature have worked really well because players can successfully undo moves and also redo undone moves. Also, when a player have made a move, players can no longer redo moves that have been previously undone. This prevents cheating. Also, moves are undo or redone in a very fast and efficient manners thanks to the array implementation of stack.

The replay feature have partially worked well because even though, It have successfully subsequentially replay each move from initial game state to final game state everytime the player clicks on a key, the ‘O’ and ‘X’ is placed same time instead of seperatly.

The replays for all games and games for the loggin player have worked partially well. Users can efficiently search up moves on a certain previous game in a efficient manner by typing in game id of a certain game and the moves are replayed subsequentially from initial game state to final game state but each ‘X’ and ‘O’ placed same time on board when a player presses a key instead of them being placed on board seperatly.

The hard version of player against computer have worked really well as players can’t cheat and is nearlly impossible to win against the computer due to the fact that computer using the minimax algorithm to look at all of possible moves and places the best move on the board when its turn ina very efficient and fast manner.

The easy version of player against computer have worked really well as players can’t cheat and computer only randomlly places piece on a empty square which causes no glitches.

The register feature works really well as the user can only type in a username that do not already exist in the system and also the password is encrypted before it is written to the file so player do not have to worry that their account can be hacked. When username already exists, a error message displayed console and the use ris returned back to the previous menu. When player successfully makes a account, a message stating that displayed on the console and the player returned back to the previous menu.

The login feature works really well as it successfully verify enetered username and password and allows user to go to the next menu. When the username and password not correct, a error message is displayed on console and the user is returned back to the previous menu.

## Personal Evaluation

I have learned how to implement a simple Tic Tac Toe game by implementing the game board by using a game board initialized with ‘1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’. I have successfully learned to allow players to make a move to a certain square of the game board until a player wins or there is a draw by while loop and if statements. I have managed to learned that a player can only win when the player has formed a horizontal, vertical or diag-onal sequence of three marks.

I have learned how to allow users to unod moves that I have been un done by using four array implementation of stacks where there is a stack for undone choices, undone marks, redone choices and redone marks. When a player undo a mark, latest placed mark comes off the board and the game go back to the previous game state. Players can do this until there are no moves on the board. Players can also redo moves that have been undone as long a player does not make a move. I have learned how to subsequentially replay moves from initial game state to final game state after the game ends by making program to loop through the moves of the undoMarks and undoChoices stack array in a efficient and quick manner. I have learnt how to write these moves to a file called ‘list.txt’ and read these moves to a linked list so players can search up moves of a certain game by typing in the game id of that certain game. I also managed to write the games moves to a separate files that have name of the player username and read it to a linked list so each player can search up a moves of certain game that the player have played previously by typing in the game Id of that game.

I have learnt how to implement a minimax algorithm which allows the computer to look through all of the possible moves and choose the most optimal move everytime is the computer turn while playing against a human player in a very efficient manner. I also managed to learn how to make a Tic Tac Toe where the computer randomly chooses a move while playing against a human player. I have managed to learn how to encrypt passwords by adding 3 to ASCII value of the entered password string and decrypt passwords by subtracting 3 from ASCII value of the entered password characters. I have managed to learn how to make users to register account by entering password and username in a way that it written to the ‘accounts.txt’ file and allowing users login by typing in correct username and password which can ne verified only when username and password is same as username and password from a line of ‘accounts.txt’.

A challenge I have faced was replaying moves as the array that was meant to hold all of the moves that placed on the game board after game finish was pointing to a memory slot that do not exist. The mythology that I have used to solve this challenge was that I have used the printf function between the for loop that been used to loop through the choices and marks of the game to subsequently replay the game moves so I can see the moves and choices printed out on screen. I have noticed that it was printing out all the seven moves and choices of the game but printing out strange symbols after this. The reason was because only seven moves have been made in that game while the for loop while making the program to loop through the array till i value was 9. I have decided to modify the for loop parameters making the for loop to loop through the undoChoices from index 0 to top value of undoChoices stack. The result of this was successfully managing to subsequentially replay moves from initial game state to final game on the game board.

A challenge that been faced while implementing this game was the program kept crashing when the user goes to the Games Replays section, the program started to crash. The methodology that I have took to overcome this challenge is by makinga function for test purposes that would read and print out all of the games marks and choices on the console. When it have successfully did that without making the program to crash, I have used the printf() function within ReadFromFileToLinkedList() to print out each choices and marks of the node after it been read from the file to a node of the linked list. Only one marks and choices been printed out on the console and then the program crashed. This made me realized that there is problem with how I am adding these data to node of linked list. After a quick google search, I have found out that I have not been setting next node pointer to new game after choices and moves of a node of the linked list is set to the choices and moves from the ‘list.txt’ file. I have set the next pointer of linked list to the new game node. The result of this was that users can now go to the Games replay section without program crashing and simply enter a game id to see the moves replays of a specific game.

The final challenge that been faced while implementing this game was in the first test of minimax algorithm used to make computer to place the most optimal move of all possible moves on the board where computer piece was ‘O’. The challenge was that when human first makes a move, the whole board start to be filled with X’s’ and then player one wins. The methodology that been taken to try overcome this challenge within the findBestMove() function, printf() function used to print out the score while anayzing all possible moves. The program was run again and the game was printing out 0 multiple times. that I have analysed the minimax(), findBestMove() and evaluate() function and compared it with the minimax algorithm pseudocode. This have made realized that there are many mistakes in these functions so I have made many modifications and ran the program again to check the scores values. The value printed out was 0 few times then -10 was printed out which shows that the computer managed to find out what is the most optimal move. The result was that the when player one makes a move, the computer looks at all possible moves and chooses the best move. This repeats until either computer or human wins the game.

Overall, I think I have successfully made a exceptional Tic Tac Toe game as