Algorithms and Data Structures Report

SET08122- Edinburgh Napier University
Ismael Souf

I. Introduction

The aim of the coursework is to demonstrate the understanding of both theory and practises in relation to the content of the Algorithms and Data Structures module. The task is to implement a text-based Tic-Tac-Toe game using the C programming language. In this coursework we are going to implement a game board to play, a recording history in order to replay the game, features such as undo/redo moves on the board.

II. Design

First of all we need to represent the game state, I decided to implement an array one dimensions of 10 elements 0 to 9. Basically the player chose a number on the board (0-9) and he/she will see a symbol appear in the number. In fact I preferred 1D array to 2D because it's easier to use in order to create the board and making the position choice for the player. When a player make a move the position is going to be quick and efficient. We also know that an array of one dimension take less memory than a 2D array. Usually a disadvantages of arrays is that we need to know how many elements are to be stored, here we know that a board of Tic-tac-toe is 3x3 so for a 1D array 9 elements are required. I especially paid attention to the fixed size of arrays in order to not waste memory neither create problem by allocating less memory.
Player 1 will have the symbol X and Player 2 the symbol 0.
Since the elements of arrays are stored in consecutive memory locations insertions and deletions are time consuming so I decided to implement stack in order to allow the game to support undo and redo function.

Figure 1. Tic-Tac-Toe Board.

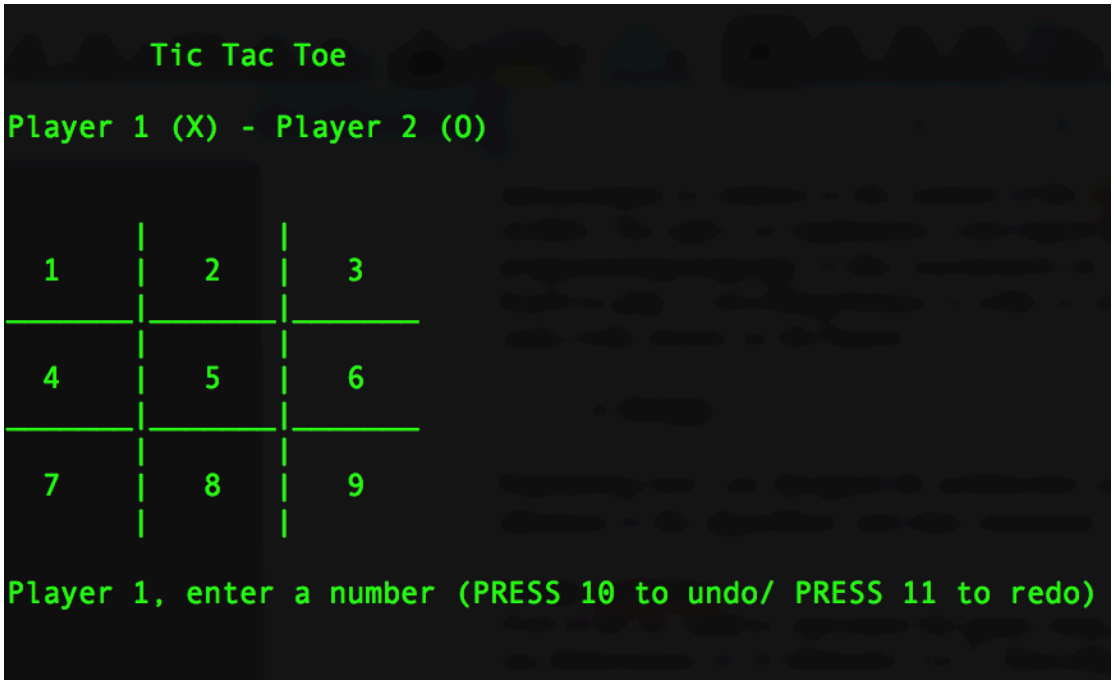


Figure 2: Undo-redo option for players

As we can see in the previous figure both players can undo their moves by pressing 10 and redo their move by pressing 11. I implemented a function for each, the undo consist of using the stack in order to remove the marks and choices made by the player during the game. Basically we need to use four stacks because in the implementation we will need to undo the choices and the symbol chosen by the player then redo the choices and the symbol. We pop from stack to remove the appropriate choice made eg. If the choice was one, the stack is going to look for the move one on the board and cancel it.
The standard is to keep the choices and symbols in stacks to support multi level undo. In order to support redo, two stacks has been implemented to keeps all the commands I have undone. So when I pop the undo stack to undo a command, you push the same command you popped into the redo stack. You do the same thing in reverse when you redo a command. You pop the redo stack and push the popped command back into the undo stack. This is because the stack is implemented with Last in First out order which is equivalent to redo-undo feature. It is very efficient to use stacks in order to use the undo-redo feature in this coursework. References are provided in the end of how I dealt with my stacks.

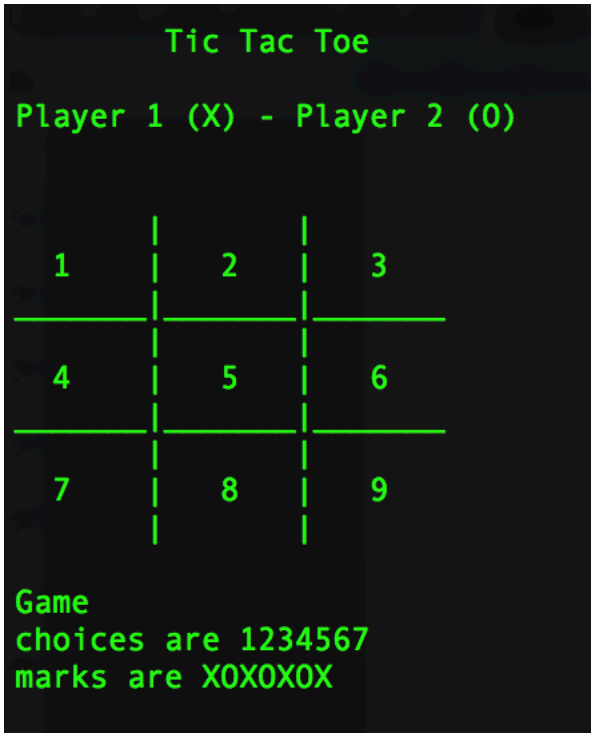


Figure 3. Replay moves of a selected game

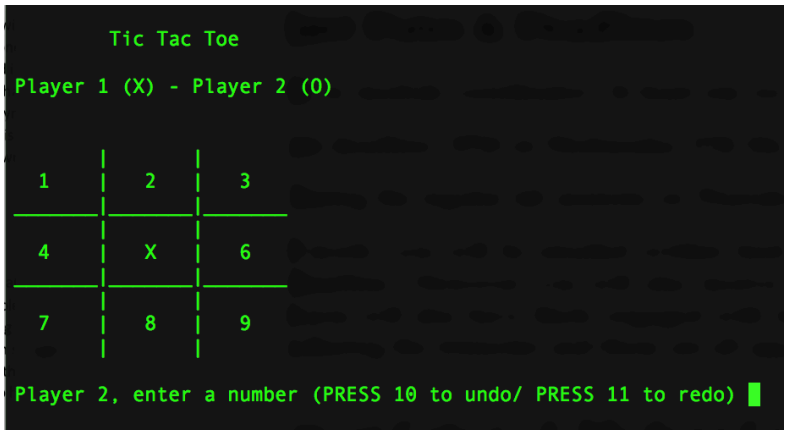
The replay feature is supposed to replay every moves of a game played by players. I decided to implement a linked list for this feature because it suits well for reading every move into a file in order to replay them. The function will read the choices and symbols chosen by players while playing after each game. Once a game has been found in "SEARCH GAME" the number of the game will be display at the top and every move will be repeated into a new board. Even though we cannot see different record game at the same time we can press a random key in order to go back to the menu and chose a game from 0. Linked list is a good choice to implement this feature because we don't need to think about the size of memory, every move is store anywhere in the memory and we don't need to think about how many games we can add into our text file.

III. Enhancements

I lacked organization for the project. I had three coursework at the same time. Otherwise I would implement an AI player. Through my searching I found that a Minimax algorithm was the most suitable to create an AI player but seemed to need time in order to figure out how I could add it to my coursework. I also thought about changing the board size, which is not difficult, but I realised at the end that my implementation would have to change from the beginning and I would have use a 2D array in order to make the feature easier. I didn't know that I could reach the 60%-69% marking scheme that's why I didn't read features in the mark 70% band.

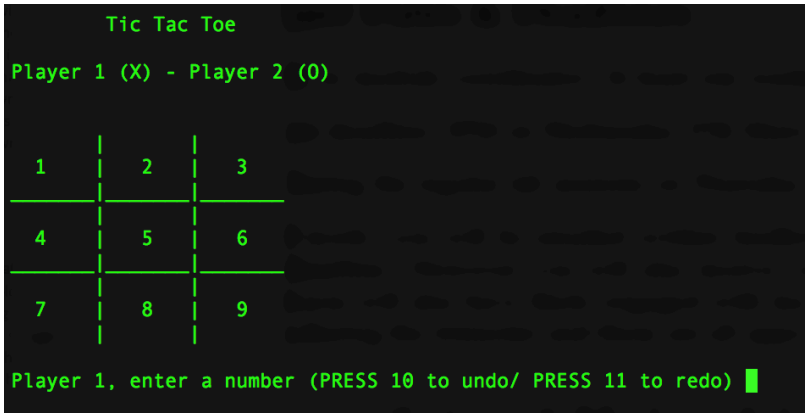
IV. Critical Evaluation

While evaluating my program I realised that I didn't know how to implement my undo and redo move so I decided to look for this features in another language to understand how to do it and then implement it in C. I think my program is well structured and contain solid function such as the stacks used in order to undo-redo moves on the board. In addition to that the undo can return to the initial game state. I found the redo easy to do once you get the undo function working. I think stack is the best way to implement those functions. Even though the program is working well I still find my code repetitive.



Player 1 chose the position 5.

Player 1 is asking to Undo his previous move.



Successful to undo the move do you want to redo? Press 11.

For the replay I don't find it perfect but I think it's doing the job asked because we can view a previous game played by players when we chose a game ID. Therefore if we don't play at least one game we are going to get a file which is empty. However every game is recorded from the start to the end and we cannot see which move has been undone and redone. If the file doesn't exist we will get a segmentation fault and the game will exit.

```
1234567 X0X0X0X
7239154 X0X0X0X
164897325 X0X0X0X0X
84132695 X0X0X0X0
162973 X0X0X0
```

Figure. Recorded Game in text file.

V. Personal Evaluation

We learnt how to program a bit in C during our first year. This coursework wasn't hard but I spent a lot of time. Through the coursework I realised that my knowledge in C wasn't that good I needed to do researches, look for additional exercises and different websites to understand a lot of things. I also used my coursework of first year in order to refresh few skills.

The challenge I faced was every feature because we had many ways to implement them. To begin with, once the board was done I wanted to follows the features recommended in the coursework specification. The replay function was easier than excepted because when I saw the word record I immediately thought about adding my moves somewhere to store and I remembered my first coursework using files. Then comes the undo-redo function they are basically the same so when the undo was done we could have a hint of how to do the redo. One of my favourite data structure is Stack followed by Trees, so I really enjoyed doing the coursework. I have to admit that a few codes that I have implemented can be found online but in another method.
Every function I have written looks repetitive but easier to be understood. That is why I think I have achieved something solid during this coursework because it improved my skills in programming.

V. Reference

Stack implementation: https://www.geeksforgeeks.org/stack-data-structure-introduction-program/

Read linked list from file:
https://stackoverflow.com/questions/44266474/read-linked-list-from-a-file-and-load-the-data

Lab SET08122-Data Structures: Contiguous Structures