



TIC TAC TOE – ASSIGNMENT

UNIX AND C PROGRAMMING (COMP1000)

BRETT STEVENS
CURTIN UNIVERSITY
Computing Discipline

Contents

1. Overall Purpose:.....	1
tictactoe.c:	1
fileReader.c:	1
UI.c	1
game.c.....	1
logs.c	1
LinkedList.c.....	1
2. Implementation of log structs and linked list:	1
3. How to:.....	2

Table of Figures

Fig 1.....	2
Fig 2.....	2
Fig 3.....	2
Fig 4.....	3
Fig 5.....	3

1. Overall Purpose:

tictactoe.c:

The tictactoe.c file includes fileReader.h and UI.h and contains only one function: the main function. The main functions purpose is to read in command line arguments, initialise memory storage for the m-n-k values and the linked list used to store logs and then pass appropriate variables into each function it calls. It also performs error checking to ensure the program doesn't run without suitable m-n-k values and frees the linked list at the end of the program.

fileReader.c:

The fileReader.c file handles file reading of the input settings file and includes fileReader.h. It scans in values from the file and handles all file error checking and error checking of each value. If errors do occur or additional elements are found in the file, it outputs appropriate error messages to the user. If errors do not occur fileReader.c saves/parses the m-n-k values to be used later in the program.

UI.c

The UI.c file (or User Input) handles all user input throughout the program; it includes UI.h, game.h and logs.h. Its main functionality is to allow the user to choose different paths to different modules with its menu and to handle each player's input for each turn during a game of tic-tac-toe. It also allows the user to edit the m-n-k values when the program has been compiled with *Editor*.

game.c

This file contains all functions that contribute to running a game of tic-tac-toe that don't require user input; it includes game.h and UI.h. The purpose of game.c is to allow for the functionality of a working tic-tac-toe sim. It handles the memory allocation of the board spaces, printing the board to screen and checking after each turn if the game is either a win or a draw. Essentially it provides each new game.

logs.c

This files purpose is to allow each move in a tic-tac-toe game to be saved as a log struct to the linked list. It is able to save multiple games to the list and print each game to screen or to file. It includes logs.h.

LinkedList.c

The LinkedList.c file contains functions for a generic linked list; it includes LinkedList.h and logs.h. Its purpose is to act as storage for the log structs of each turn of each game. It also plays a big part in preventing memory leaks as one of its functions iterates through the linked list freeing each log at the end of the program.

2. Implementation of log structs and linked list:

Logs of each turn in each game are stored to then be later printed to screen or saved to file. Each log is a struct that contains information such as which turn it is, the player who took the turn and the location of the character placed. By using a generic linked list, we are able to store all of this information easily and, just as importantly, free the information when needed. We do this by using a function pointer.

The only complication faced when designing this section of code was how to print/save the logs in

the specified format: with the game number displayed and three dots breaking up each game as seen in *fig 5*. This problem was solved by handling the iteration used for the printing/saving within the logs.c file, instead of using a generic print linked list function.

3. How to:

The m-n-k tic-tac-toe program is a simple simulator that allows you to play a game of tic-tac-toe (or naughts and crosses for those of us still clinging to the old ways) on a board of 'm' length and 'n' height, with the winner determined by the first player that gets 'k' amount of naughts or crosses in a row. The program inputs a file that contains the games settings, an example of such a file can be seen in *fig 1* below:

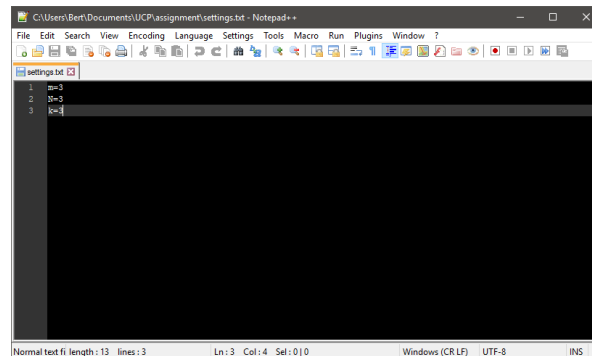


Fig 1

Here are a few simple steps of how to use the program:

1. Command line

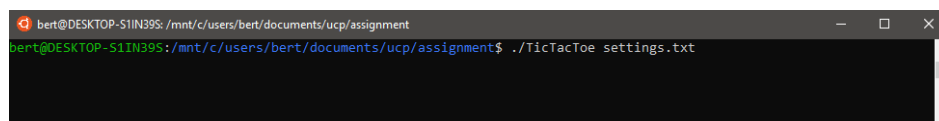


Fig 2

Fig 2 shows an example of the command line used to execute the program. Command line is of the form: `./TicTacToe <settings file to be used>`

2. Menu

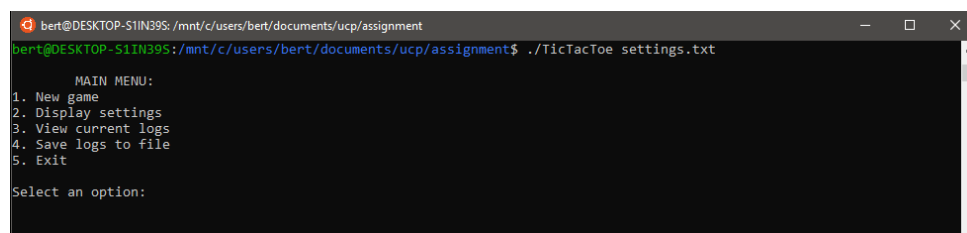
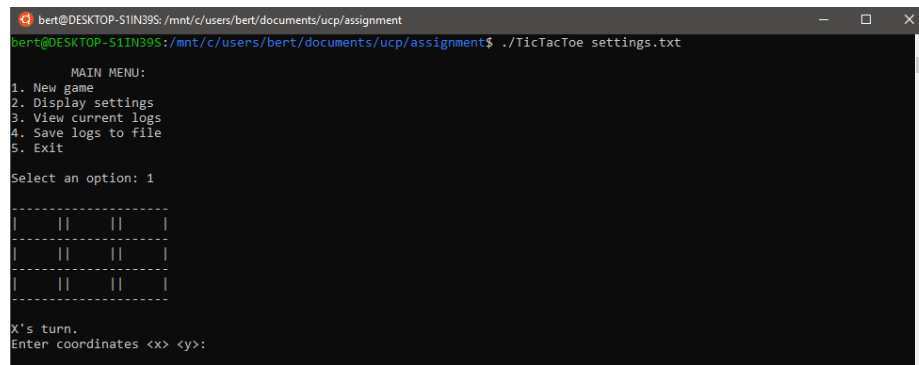


Fig 3

Once the program is executed the user will be prompted to select an option from the menu. To select one of the options the user must input the appropriate number and hit enter. To select new game simply input 1 and hit enter.

3. New game



```
bert@DESKTOP-S1IN39S: /mnt/c/users/bert/documents/ucp/assignment
bert@DESKTOP-S1IN39S:/mnt/c/users/bert/documents/ucp/assignment$ ./TicTacToe settings.txt

MAIN MENU:
1. New game
2. Display settings
3. View current logs
4. Save logs to file
5. Exit

Select an option: 1

-----
|  |  |  |
|  |  |  |
|  |  |  |
-----

X's turn.
Enter coordinates <x> <y>:
```

Fig 4

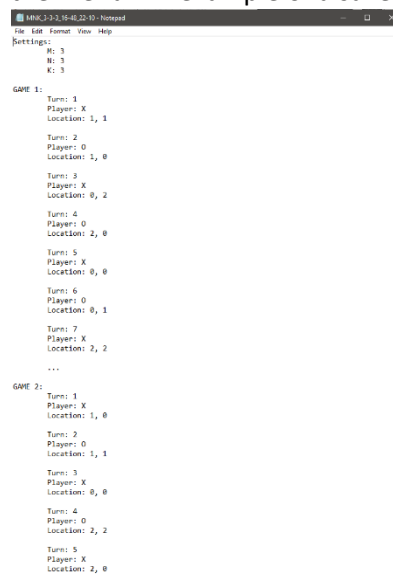
When new game is selected the program will print out the board specified by the m-n values found in the settings file. The program will also prompt crosses, as is tradition, to go first.

To select a position on the board the player must input two numbers separated by a space. This input represents the placements x-y coordinates on the board, i.e. x input is how many spaces right from the top left square you want to place your character and the y input is how many spaces down from the top left square you want to place your character; for example, the middle position in the 3x3 board seen in fig 4 above would be selected by an input *1 1*.

The program will print a new table with the appropriate character in its place and prompt the next player to enter the coordinates for their turn. This will continue until one player wins by getting 'k' amount of their characters in a row or the game ends in a draw. Once the game has ended the program will return to the main menu allowing the user to input another selection.

4. Saving logs to file

Whilst playing a game the program keeps track of all moves and can either output these logs to screen or save them to a file from the menu. An example of a saved log file can be seen in fig 5 below:



```
Settings:
R: 3
M: 3
K: 3

GAME 1:
Turn: 1
Player: X
Location: 1, 1
Turn: 2
Player: O
Location: 2, 0
Turn: 3
Player: X
Location: 0, 2
Turn: 4
Player: O
Location: 2, 0
Turn: 5
Player: X
Location: 0, 0
Turn: 6
Player: O
Location: 0, 1
Turn: 7
Player: X
Location: 2, 2
...

GAME 2:
Turn: 1
Player: X
Location: 1, 0
Turn: 2
Player: O
Location: 1, 1
Turn: 3
Player: X
Location: 0, 0
Turn: 4
Player: O
Location: 2, 2
Turn: 5
Player: X
Location: 2, 0
```

Fig 5

5. Conditional compilation

To obtain an additional edit function to edit the m-n-k values in the menu define *Editor* when compiling (i.e. use *make Editor=1* instead of *make* to compile). To remove the ability to save logs define *Secret* when compiling (i.e. use *make Secret=1* instead of *make* to compile).