PROGRAMMING ASSIGNMENT 5

Subject : Dynamic Memory Allocation, Structures
Due Date : 31.12.2014

Introduction. In this experiment, you will implement a simple board game that is played on a board structured as a grid. Main focus of this experiment is to get you familiar with dynamic memory allocation and structures. Implementing the program without utilizing dynamic memory allocation and/or structures will be penalized.

Board Games

A board game is a game that involves counters or pieces moved or placed on a pre-marked surface or "board", according to a set of rules. Games can be based on pure strategy, chance (e.g. rolling dice), or a mixture of the two, and usually have a goal that a player aims to achieve. Early board games represented a battle between two armies, and most modern board games are still based on defeating opposing players in terms of counters, winning position, or accrual of points (often expressed as in-game currency).

Assignment

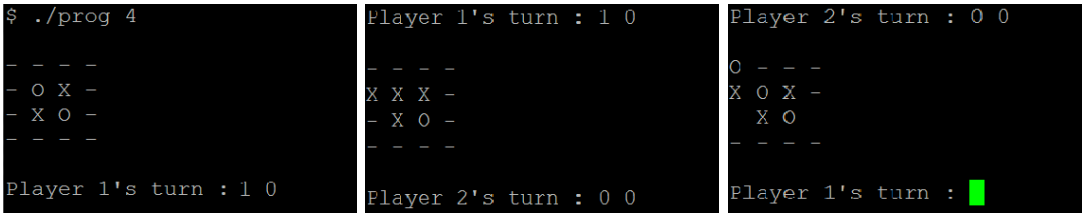
Board and Pieces

The game you will implement is a two player board game that will be played on a board structured as a square grid. Size of the grid will be provided in advance when the application is started. Size should greater then or equal to 4 and should always be an even number. The game is played via placing "pieces" on the cells of the grid. Each piece is coloured as white at one side and black at the other one representing the colours of each player.

Game Rules

The game starts with 4 pieces placed on 4 cells at the center. Two of the pieces should be placed as the white face up (character 'O' in this experiment) while other two should be placed as the black face up (character 'X' in this experiment). The game is played in turns and the black player starts first. Each player -in their turn- place a piece on one of the empty cells of the grid with the corresponding face up. See Figure 1.

Main rule of the game is that when a player places a piece on the board, it must capture other player's pieces. This means that the new piece should be placed at the end of an array of (diagonal, vertical or horizontal) other player's pieces and the same array should also be covered by another friendly piece already on the table at the other side. SEE EXAMPLES. The captured pieces will be flipped in order to represent the capturer's colour. A player



(a) (b) (c)

Figure 1: Initial stages of a game.

cannot make a move in their turn if there are no possible moves to capture other player's pieces. If the board is fulled or neither of the players can make a valid move, the game ends.

When a piece is placed on the board (including the initially placed 4 pieces) they worth 1 point. After that, value a piece is increased for the first 4 times it was flipped. The increment amount is equal to the number of flipped pieces at that turn. For example, if a player captures 3 pieces at once, each of those pieces' values increase by 3 provided that they have been flipped less then 4 times before.

When the game is finished, two values are calculated for each player that are :

1. Number of pieces controlled on the board
2. Total points earned from the controlled pieces

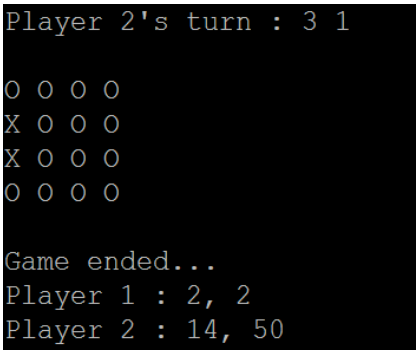


Figure 2: End of a game.

User Interface

The game will be played via the console. When the application is started, initial state of the game board will be displayed. After that, for each turn, user will be asked for a coordinate as the next players move. The user should enter the coordinates with a space between them and then press enter. When the game is finished (either by filling the entire board or running out of valid moves for both players) the score will be displayed in the console.

The size of the game board will be supplied via a command line argument. The size argument will be one and only extra argument.

Error Handling

When an erroneous situation occurs, the application should provide a meaningful error message. Below you can find sample error messages for the faulty situations. If you identify any other situation where the normal execution of the program cannot continue, you can provide additional error messages of your own.

- *Error : Invalid number of command line arguments.*
  This error message should be printed if the number of command line arguments is not equal to the expected value. Execution of the program should terminate in this situation.
- *Error : Invalid format for command line arguments. The size argument should be an even positive integer number greater then or equal to 4.*
  This error message should be printed if the size argument is not a positive even integer number greater then or equal to 4. Execution of the program should terminate in this situation.
- *Error : Invalid move. Please make another move :*
  This error message should be printed if the user enters invalid coordinates as the next move (i.e. somewhere outside the grid or somewhere already occupied by another piece). This time, the execution should continue by taking another coordinate from the user.

Design Notes

- You are expected to develop the program on the departments servers (at dev.cs.hacettepe.edu.tr). If somehow you cannot develop your program on the servers, at least make sure that you test it on the servers before you submit your work. Your submission will be evaluated on the department servers. It is your responsibility to make sure that the source code you submit successfully builds and runs on the department servers.
- You should obey general programming principles. Function and variable names should be meaningful and should provide information on the usage. You should provide comment lines for the code parts that you think is important and hard to understand. Also, using global variables is not permitted.
- Divide the program into functions that are well defined and capable of performing micro tasks. Later you can combine those functions into higher level ones that use them. Writing all the code in only a few functions will be penalized.
- Examine the provided examples carefully and make sure that your program follows the input-output format of these samples. Keep in mind that a program that does not work 100% right is a program that works wrong.

Assignment Report

Your final report should contain these topics:

1. Cover Page
2. Software Design Notes
   (a) Problem: Re-define the experiment in your own words. Keep the definition short by mentioning only the important parts. DO NOT COPY PASTE FROM THIS DOCUMENT.
   (b) Solution: Explain how you approached the problem, what was the most important part of the problem from your perspective and how it effected your code. And then add anything else that you think will further explain your design.
   (c) Explain main data structure(s) you used in your code.
3. Error Messages: Describe additional error messages (if any) that is included in the software explaining its possible causes.

Notes and Restrictions

- Do not submit any file via e-mail. You should upload your files via Online Experiment Submission System which is at `http://submit.cs.hacettepe.edu.tr`
- Submission Format:
  - <student_number>.zip
    - src
      - hw5.c
    - report
      - report.pdf
- You should submit only the source code. Do not submit the executable file.
- Save all your work until the assignment is graded.
- This is not a team project. You are expected to provide an **INDIVIDUAL** work.
- You should follow announcements on BBM103 page of piazza.
- Late Submission Policy: You may use up to three extension days for the assignment. But each extension day will bring about additional 10% degradation for evaluation of the assignment.