CS2461-10 Computer Architecture I
Spring 2016

Project 2 03/28/2016

Due 04/28/2016

This project is to practice the C programming skills. Below is a list of topics for your selection. This is a team-work project, and at most two students are allowed to work together. The grading is based on the policy:
60% correct results, 20% program clarity and comments, 20% robustness.

1: Searching and Replacing Substring

Using the example in Chapter 13 as the starting point, this program is to search and replace the user input word in a given text file. Given a text file in the same location of your program, prompt user to input a query word and a replacement word. Your program searches the text file for the query word and replace it with the second input word. After reaching the end of the text file, your program reports the number of replacement, and save the text file.

2: TicTacToe

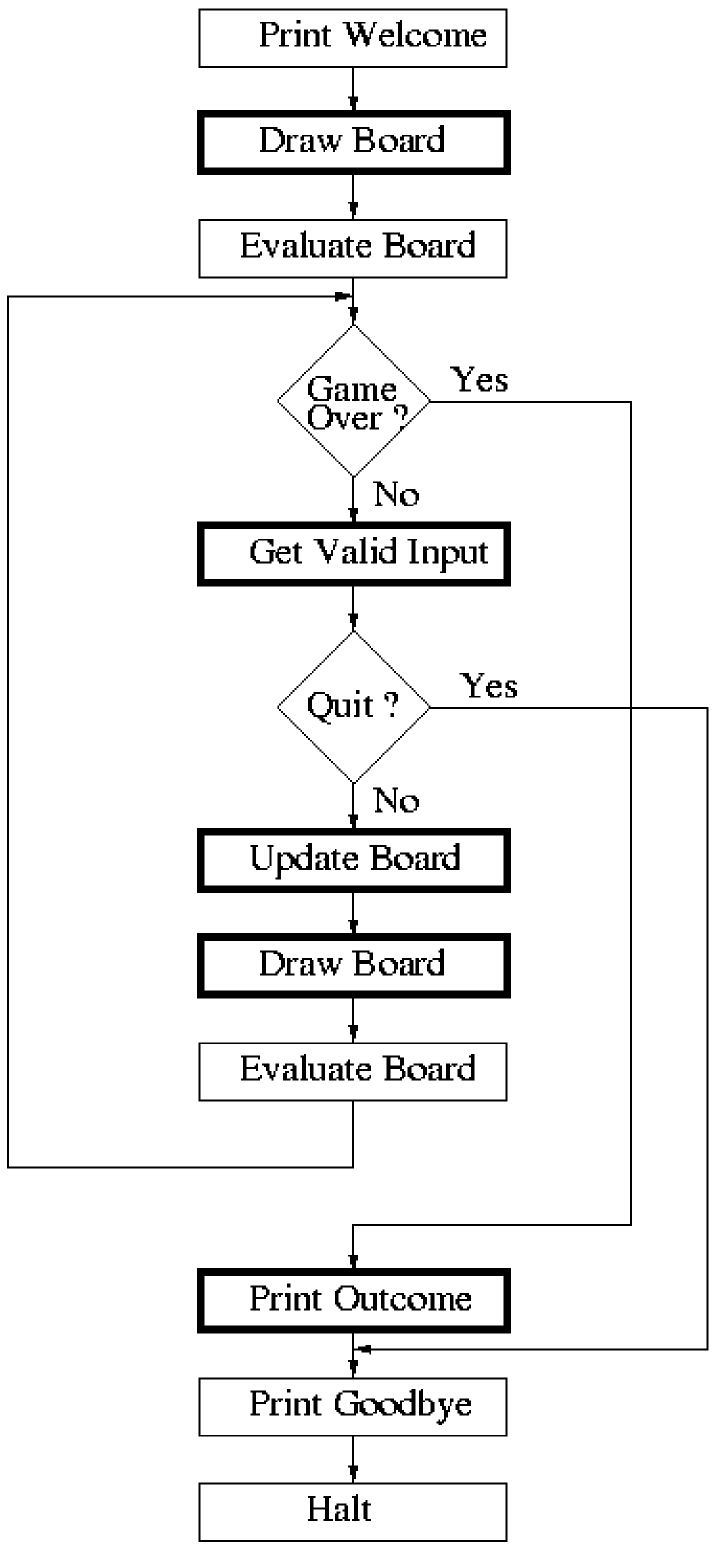
Introduction: This program is to implement the game of Tic-Tac-Toe. This simple two-player game has even been used as the basis for a popular television show called Hollywood Squares. In this assignment, you will use the principles of subroutine call and return. Using the high-level flow chart given in section 3, you will implement an ASCII Tic-Tac-Toe game for the LC-3.

Rules: Tic-Tac-Toe is a two-player game with two symbols denoting the two players: X and O. The playing board is a 3x3 square.

The rules are as follows:

- Each player takes a turn placing his character (X or O) into one of the nine squares.
- A player cannot place his symbol in a square that is already occupied by a symbol.
- The game ends when a player creates a winning combination of his symbols or when there are no empty squares remaining.
- Winning combination is defined as three horizontally adjacent, three vertically adjacent, or three diagonally adjacent symbols.
- If neither player creates a winning combination when all nine squares are occupied, the game is a draw, often referred to as a "cat game."

Algorithm: In order to design a Tic-Tac-Toe game, or any other software project, we must break the problem into small pieces. To this effect, we have created the following flow chart for Tic-Tac-Toe.



Function Specifics:

- GETVALIDINPUT
  This subroutine combines two tasks:
  1. Getting input from the user (this input is returned to the caller in register R1)
  2. Performing error checking on the input

  The only valid input during the Tic-Tac-Toe game is a number 1-9, or the letter 'q'.
  You should prompt the player for input with the following prompt:

  ```
  "X - Which square? [1-9] "
  "O - Which square? [1-9] "
  ```

  If the player does not enter a valid input, you should print the following message to the screen

  ```
  "Invalid input"
  ```

  and again prompt the player to choose a square.

  If the player chooses a square which is already occupied, you should print the message

  ```
  "Space occupied"
  ```

  and again prompt the player to choose a square.

  The ASCII code of the valid input character must be returned in a dedicated variable IN.

- UPDATEBOARD
  When this subroutine is called, the variable IN contains the ASCII code for a numeric character 1-9, which you previously have provided in GETVALIDINPUT.
  The player, X or O, is defined by the contents of the memory location labeled PLAYER, which is initially set to 1.

  ```
  Memory[PLAYER] = 1 for player X
                   0 for player O
  ```

  You should update the Tic-Tac-Toe board in memory by writing an X or O into the square designated by the value in IN. You must also update the contents of location PLAYER after updating the board.

- DRAWBOARD
  This subroutine simply prints the Tic-Tac-Toe board to the screen.

- PRINTOUTCOME
  When this subroutine is called, a declared variable OUT contains a value indicating the outcome of the game. This value is provided by the EVALBOARD subroutine.
  1 = X wins
  2 = O wins

3 = Cat Game
You should print one of the following messages based on the value in R1.

```
"X wins!"
"O wins!"
"Cat Game!"
```

- EVALBOARD
  This code examines the nine squares and returns a value in register R1 according to the following:

```
R1 = 0     Game still in progress
     1     X wins
     2     O wins
     3     Cat Game (no winner)
```

Guidelines: You should adhere to the following guidelines when designing your program.

- The squares on the board are numbered sequentially, starting from the top-left corner as shown below:

```
   |   |
 1 | 2 | 3
   |   |
-----------
   |   |
 4 | 5 | 6
   |   |
-----------
   |   |
 7 | 8 | 9
   |   |
```

- X always goes first. This is especially important for grading!

Example: Remember, X always goes first. You may test your game with the following inputs:

```
InputString          OutputString
---------------      ------------
1 4 2 5 7 6          O wins!
5 4 3 1 7            X wins!
1 2 3 7 8 9 4 5 6    Cat Game!
```

3: EASY as ABC

Introduction: Easy as ABC is a puzzle game by Wei-Hwa Huang. In the puzzle the given letters (A, B, C) are placed in the grid so that each letter appears just once in each row and column. Further, letters are given outside the grid. These letters tell that letter will be the first found in that row or column starting from that direction. In addition some cells in the puzzle will be filled (X) at the start of the game. The game is played on a 4x4 grid. The grid uses cells 8-11, 14-17, 20-23, and 26-29. The other cells are used to place first found letters. Cells 1, 6, 31, and 36 are not used.

Grid #1

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

The outside A in Grid #2 is at location #7. The Grid #3 is the output result.

Grid #2

A		X			
				X	C
C			X		
	X				
	C				

Grid #3

A	X	B	C
B	A	C	X
C	B	X	A
X	C	A	B

Specifics:

Input: there will be 5 lines of input. Each line will contain the location number of 4 filled cells. That will be followed by the number of first found letters given and their value and location. That will be followed by a cell location. The input data for Grid #2 is given in Sample Input #1.

Output: for each input line print the letter that correctly fills the given cell location. In Sample Input #1, cells 9, 17, 22, and 26 are filled. 4 first found letters and locations are given. The last location on the line, 14, is the cell to be filled by the correct letter.

Sample Input:	**Sample Output:**
#1. 9, 17, 22, 26, 4, A, 7, C, 18, C, 19, C, 32, 14	1. B
#2. 11, 16, 20, 27, 4, A, 7, B, 19, A, 24, B, 30, 22	2. C
#3. 9, 14, 23, 28, 3, B, 7, A, 19, A, 30, 10	3. A
#4. 8, 15, 23, 28, 4, A, 7, C, 24, C, 33, A, 30, 20	4. A
#5. 9, 16, 23, 26, 4, A, 7, B, 19, B, 25, B, 18, 15	5. A