CS 131 Assignment 9
Tic-Tae-Toe

Note: This is a group assignment with one to three students in a group. The group leader shall be responsible for uploading the deliverable items.

Objectives: To learn how to program in the LC-3 assembly language, including:

- subroutines
- examine complex program

1. Introduction

Ever had a couple of minutes to spare? We all have. Chances are, during many such moments as a child, you decided to play a game. One of the most popular such "time filler" games is 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. In section 4 we provide you with some assembly code to help you get started.

2. Rules

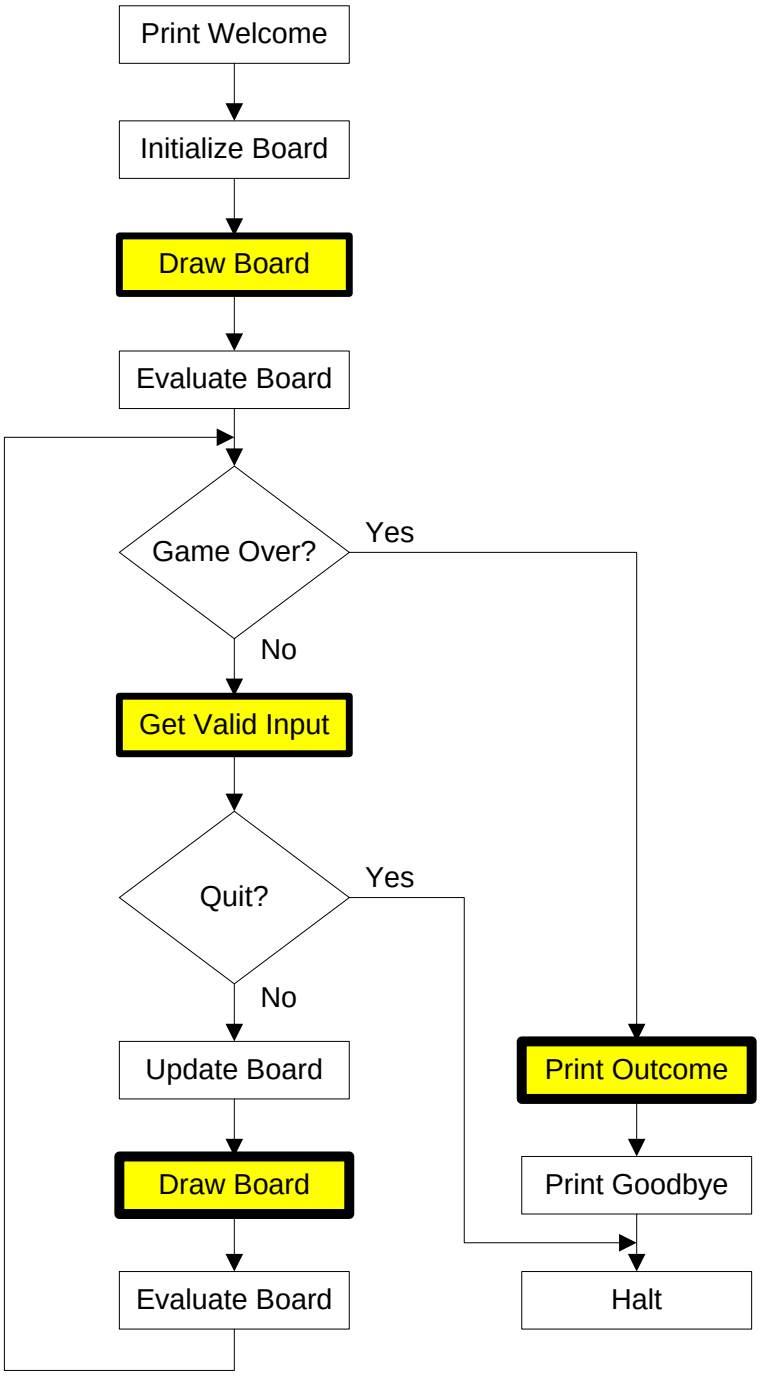
Tic-Tac-Toe is a two-player game with two symbols denoting the two players: X and O. The playing board is a 3x3 square.

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.
- A 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."

3. 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. You will use this flowchart as the basis for designing the program. The blocks that you must implement are highlighted yellow.



4. What to Do

Download the **tictactoe.asm** starter code from Canvas. This code provides a general framework for the Tic-Tac-Toe game, plus a few pre-coded internal subroutines. Your job is to complete the Tic-Tac-Toe game by writing the following three subroutines.

4.1 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, letter 'q', or the letter 'x'. You should prompt the player for input with the following prompt:

```
"X - Which square? [1-9,q,x]: "
"O - Which square? [1-9,q,x]: "
```

Input ‘q’ quits the current game whereas ‘x’ exits the program (executes a HALT) 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 (note: this code has been provided for you).

The ASCII code of the valid input character must be returned in register R1.

4.2 DRAWBOARD

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

4.3 PRINTOUTCOME

When this subroutine is called, R1 contains a value indicating the outcome of the game.

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!"
```

4.4 EVALBOARD (provided)

The EVALBOARD subroutine is provided for you and evaluates the playing board. 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)

Examining the EVALBOARD subroutine will provide insight into LC-3 assembly language programming.

5. 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!

- In each subroutine you write, you should save and restore any registers that you use. This will avoid a major headache during debugging.

- **Hint**: use code from the previous assignments to help you with the inputs and outputs on this assignment.

6. Deliverables

Run your program with the following inputs (user inputs are **bold**):

Input String	Output String
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!

After running your program have your Group Leader submit the following files on Canvas:

1. A screen capture of your program's output (the console window) for the **ENTIRE LAST** test case (**the Cat Game**) – see below.

2. Your source code (`tictactoe.asm` file).

Example console output: