**TIC TAC TOE**PF lab project in C

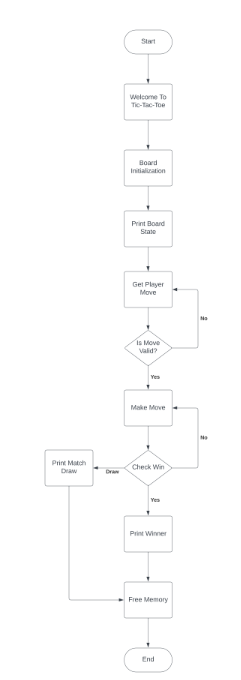
By: Ahmad Saif Ullah, Muhammad Hammad Azim

**Algorithm:**Input: None  
Output: None  
  
1. Print "WELCOME TO TIKI TAKI TOE"  
  
2. Initialize the game board:  
   2.1. Allocate memory for a 3x3 board.  
   2.2. Initialize each cell of the board to ' '.  
  
3. Set currentPlayer to 'X'.  
  
4. Repeat the following steps until the game is over:  
   4.1. Print the current state of the board.  
   4.2. Get the player move (row, col).  
   4.3. If the move is invalid (out of bounds or the cell is already occupied), print "INVALID MOVE!! Try again." and go to step 4.2.  
   4.4. Make the move on the board.  
   4.5. If the current player has won, print "Player X (or O) wins!" and end the game.  
   4.6. If the board is full, print "The game is a tie!" and end the game.  
   4.7. Switch to the other player.  
  
5. Free the memory allocated for the board.

**Pseudocode:**

// Function prototypes  
void displayBoard(char \*\*board);  
int checkWin(char \*\*board, char player);  
int isBoardFull(char \*\*board);  
void freeBoard(char \*\*board);  
  
// Main function  
function main():  
    print("WELCOME TO TIKI TAKI TOE")  
  
    // Dynamic allocation of the Tic-Tac-Toe board  
    board = allocateMemoryForBoard()  
  
    currentPlayer = 'X'  
  
    repeat:  
        printBoardGuide()  
        displayBoard(board)  
  
        // Get player move  
        print("Player", currentPlayer, "enter your move (row and column): ")  
        row, col = getPlayerMove()  
  
        // Check if the move is valid  
        if isInvalidMove(row, col, board):  
            print("INVALID MOVE!!\nTry again.")  
            continue  
  
        // Make the move  
        makeMove(board, row, col, currentPlayer)  
  
        // Check for a win  
        if checkWin(board, currentPlayer):  
            displayBoard(board)  
            print("Player", currentPlayer, "wins!")  
            break  
  
        // Check for a tie  
        if isBoardFull(board):  
            displayBoard(board)  
            print("The game is a tie!")  
            break  
  
        // Switch to the other player  
        currentPlayer = switchPlayer(currentPlayer)  
  
    until false  
  
    // Free dynamically allocated memory for the board  
    freeBoard(board)

**Flowchart:**

****

**Code:**

#include <stdio.h>  
#include <stdlib.h>  
  
// Function prototypes  
void displayBoard(char \*\*board);  
int checkWin(char \*\*board, char player);  
int isBoardFull(char \*\*board);  
void freeBoard(char \*\*board);  
  
int main() {  
    printf("WELCOME TO TIC-TAC-TOE\n");  
  
    // Dynamic allocation of the Tic-Tac-Toe board  
    char \*board = (char \*)malloc(3 \* sizeof(char \*));  
    for (int i = 0; i < 3; i++) {  
        board[i] = (char \*)malloc(3 \* sizeof(char));  
        for (int j = 0; j < 3; j++) {  
            board[i][j] = ' ';  
        }  
    }  
  
    int row, col;  
    char currentPlayer = 'O';  
  
    do {  
        printf("\n 0,0 | 0,1 | 0,2\n---------------\n 1,0 | 1,1 | 1,2\n---------------\n 2,0 | 2,1 | 2,2\n");  
  
        // Display the current board  
        displayBoard(board);  
  
        // Get player move  
        printf("Player %c, Enter Your Move (Row and Column): ", currentPlayer);  
        scanf("%d %d", &row, &col);  
  
        // Check if the move is valid  
        if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != ' ') {  
            printf("\n\nINVALID MOVE!!\nTry Again.\n");  
            continue;  
        }  
  
        // Make the move  
        board[row][col] = currentPlayer;  
  
        // Check for a win  
        if (checkWin(board, currentPlayer)) {  
            displayBoard(board);  
            printf("Player %c Wins!\n", currentPlayer);  
            break;  
        }  
  
        // Check for a tie  
        if (isBoardFull(board)) {  
            displayBoard(board);  
            printf("The Game Is A Tie!\n");  
            break;  
        }  
  
        // Switch to the other player  
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';  
  
    } while (1);  
  
    // Free dynamically allocated memory for the board  
    freeBoard(board);  
  
    return 0;  
}  
  
// Function to display the Tic-Tac-Toe board  
void displayBoard(char \*\*board) {  
    printf("\n");  
    for (int i = 0; i < 3; i++) {  
        for (int j = 0; j < 3; j++) {  
            printf(" %c ", board[i][j]);  
            if (j < 2) printf("|");  
        }  
        printf("\n");  
        if (i < 2) printf("-----------\n");  
    }  
    printf("\n");  
}  
  
// Function to check if a player has won  
int checkWin(char \*\*board, char player) {  
    // Check rows and columns  
    for (int i = 0; i < 3; i++) {  
        if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) ||  
            (board[0][i] == player && board[1][i] == player && board[2][i] == player)) {  
            return 1; // Player wins  
        }  
    }  
  
    // Check diagonals  
    if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) ||  
        (board[0][2] == player && board[1][1] == player && board[2][0] == player)) {  
        return 1; // Player wins  
    }  
  
    return 0; // No winner yet  
}  
  
// Function to check if the board is full  
int isBoardFull(char \*\*board) {  
    for (int i = 0; i < 3; i++) {  
        for (int j = 0; j < 3; j++) {  
            if (board[i][j] == ' ') {  
                return 0; // Board is not full  
            }  
        }  
    }  
    return 1; // Board is full  
}  
  
// Function to free dynamically allocated memory for the board  
void freeBoard(char \*\*board) {  
    for (int i = 0; i < 3; i++) {  
        free(board[i]);  
    }  
    free(board);  
}