DATA STRUCTURE II COURSE PROJECT REPORT

TIC-TAC-TOE GAME

Project Submitted by:

Akhilesh Vallabhaneni

AP22110011504

B.Tech CSE-A

SRM University AP

1. PROBLEM DEFINITION

This Project was about the Tic-Tac-Toe game provided in simple C Language using Data structures. This program functions as it consists of two players. In which their turns will be shuffling each time. They need to mark their symbols accordingly. By that, we need to check whether the game was won by a particular person or the game went to a draw.

2. BRIEF DESCRIPTION OF THE SOLUTION

In this project, the two players choose the available empty spaces in the board and fill them with their symbols accordingly, this Program consists of functions such as initializing the board to empty spaces at the start, checking for the win when the respective symbol is placed or not, printing the board each time, checking whether the game came to draw or not and play game function to continue the game further until is draw.

3. SOURCE CODE (IN C)

```
#include  <stdio.h>

#include  <stdbool.h>


#define SIZE 3  //defining the variable SIZE with value 3


// Structure to represent a player

struct Player {

    char symbol;

    char name[20];

};


// Structure to represent a Tic-Tac-Toe board

struct TicTacToe {

    char board[SIZE][SIZE];

    struct Player player1;

    struct Player player2;

};


// Function to initialize the Tic-Tac-Toe board to empty spaces at starting

void initializeBoard(struct TicTacToe* game) {

    for (int i = 0; i < SIZE; i++) {

        for (int j = 0; j < SIZE; j++) {
            game->board[i][j] = ' ';
        }
    }
}

// Function to print the Tic-Tac-Toe board each time after choosing a symbol by the each player
void printBoard(const struct TicTacToe* game) {
    printf("\n");
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf(" %c ", game->board[i][j]);
            if (j < SIZE - 1) printf("|");
        }
        printf("\n");
        if (i < SIZE - 1) printf("-----------\n");
    }
    printf("\n");
}

// Function to check if a player has won or not
bool checkWin(const struct TicTacToe* game, char symbol) {
    // Check rows and columns
    for (int i = 0; i < SIZE; i++) {
```

```
if ((game->board[i][0] == symbol && game->board[i][1] == symbol && game->board[i][2] ==
symbol) || (game->board[0][i] == symbol && game->board[1][i] == symbol &&
game->board[2][i] == symbol)) {

            return true;

        }

    }



    // Checking diagonals whether they are same or not

     if ((game->board[0][0] == symbol && game->board[1][1] == symbol && game->board[2][2]
== symbol) || (game->board[0][2] == symbol && game->board[1][1] == symbol &&
game->board[2][0] == symbol)) {

        return true;

    }



    return false;

}



// Function to check if the board is full (i.e., a tie or draw)

bool checkTie(const struct TicTacToe* game) {

    for (int i = 0; i < SIZE; i++) {

        for (int j = 0; j < SIZE; j++) {

            if (game->board[i][j] == ' ') {

                return false;

            }

        }
    }
    return true;
}

// Function to make a successful move
void makeMove(struct TicTacToe* game, int row, int col, char symbol) {
    game->board[row][col] = symbol;
}

// Function to play the Tic-Tac-Toe game which continues till the end
void playGame(struct TicTacToe* game) {
    initializeBoard(game);
    printf("Enter the name of Player 1 (X): ");
    scanf("%s", game->player1.name);
    game->player1.symbol = 'X';

    printf("Enter the name of Player 2 (O): ");
    scanf("%s", game->player2.name);
    game->player2.symbol = 'O';

    int row, col;
    bool player1Turn = true;
    bool gameWon = false;
    do {
        // Print the current board
        printBoard(game);

        // Get the current player's move
        if (player1Turn) {
            printf("%s's turn (X): Enter row and column (1-3): ", game->player1.name);
        } else {
            printf("%s's turn (O): Enter row and column (1-3): ", game->player2.name);
        }
        scanf("%d %d", &row, &col);
        row--; // Adjust to 0-based index
        col--;

        // Check if the chosen cell is valid or not and to make the move
    if (row >= 0 && row < SIZE && col >= 0 && col < SIZE && game->board[row][col] == ' ')
{
            char currentSymbol = (player1Turn) ? game->player1.symbol : game->player2.symbol;
            makeMove(game, row, col, currentSymbol);

            // Check if the current player has won or not
            if (checkWin(game, currentSymbol)) {
                gameWon = true;
                printf("\n%s wins!\n", (player1Turn) ? game->player1.name : game->player2.name);
            } else if (checkTie(game)) {
                // Check for a tie
                printf("\nIt's a tie!\n");
                break;
            }

            // Switching to the other player's turn
            player1Turn = !player1Turn;
        } else {
            printf("Invalid move. Try again.\n");
        }
    } while (!gameWon);
    // Print the final board
    printBoard(game);
}
int main() {
    struct TicTacToe game;
    playGame(&game);
    return 0;
}
```

4. SAMPLE OUTPUT

- **When it is tie:**

```
Enter the name of Player 1 (X): ABC

Enter the name of Player 2 (O): XYZ

  |  |

-----------

  |  |

-----------

  |  |


ABC's turn (X): Enter row and column (1-3): 1 1

 X |  |

-----------

  |  |

-----------

  |  |

XYZ's turn (O): Enter row and column (1-3): 2 2

 X |  |

-----------

  | O |

-----------

  |  |
ABC's turn (X): Enter row and column (1-3): 3 1

 X |   |
-----------
   | O |
-----------
 X |   |

XYZ's turn (O): Enter row and column (1-3): 2 1

 X |   |
-----------
 O | O |
-----------
 X |   |

ABC's turn (X): Enter row and column (1-3): 2 3

 X |   |
-----------
 O | O | X
-----------
 X |   |
XYZ's turn (O): Enter row and column (1-3): 1 2


 X | O |

-----------

 O | O | X

-----------

 X |   |

ABC's turn (X): Enter row and column (1-3): 3 2

 X | O |

-----------

 O | O | X

-----------

 X | X |

XYZ's turn (O): Enter row and column (1-3): 3 3

 X | O |

-----------

 O | O | X

-----------

 X | X | O


ABC's turn (X): Enter row and column (1-3): 1 3
```

```
It's a tie!

 X | O | X

-----------

 O | O | X

-----------

 X | X | O
```

- **When a player wins:**

```
Enter the name of Player 1 (X): ABC

Enter the name of Player 2 (O): XYZ

   |   |

-----------

   |   |

-----------

   |   |

ABC's turn (X): Enter row and column (1-3): 2 2

   |   |

-----------

   | X |

-----------

   |   |

XYZ's turn (O): Enter row and column (1-3): 3 3

   |   |
-----------
   | X |
-----------
   |   | O
ABC's turn (X): Enter row and column (1-3): 3 2
   |   |
-----------
   | X |
-----------
   | X | O
XYZ's turn (O): Enter row and column (1-3): 1 2
   | O |
-----------
   | X |
-----------
   | X | O
ABC's turn (X): Enter row and column (1-3): 3 1
   | O |
-----------
   | X |
-----------
 X | X | O
```

```
XYZ's turn (O): Enter row and column (1-3): 1 3

   | O | O

-----------

   | X |

-----------

 X | X | O


ABC's turn (X): Enter row and column (1-3): 2 3

   | O | O

-----------

   | X | X

-----------

 X | X | O


XYZ's turn (O): Enter row and column (1-3): 1 1

XYZ wins!


 O | O | O

-----------

   | X | X

-----------

 X | X | O
```