School of Mechanical & Manufacturing Engineering (SMME),

National University of Science and Technology (NUST),

Sector H-12, Islamabad

Program: **BE-Aerospace**

Section: **AE-01**

Course Title: **Fundamentals of Programming**

PROJECT REPORT

Tic Tac Toe

Student 1:

Name: **FIZA ALI**

CMS: **454685**

Student 2:

Name: **IZBAL UMAID**

CMS: **458950**

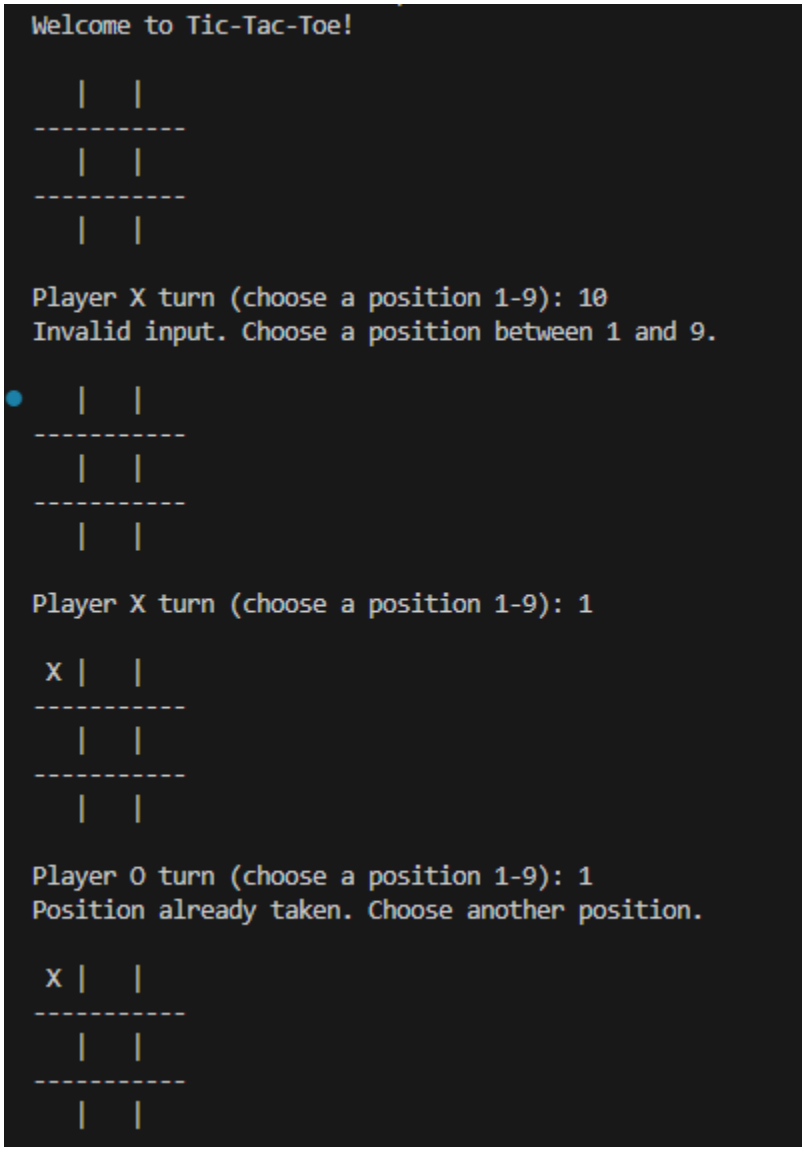
PROGRAM:

```
#include <iostream>
using namespace std;

void printBoard(char board[3][3]);
bool checkWin(char board[3][3]);
bool placeMarker(char board[3][3], int position, char marker);

int main() {
    char board[3][3] = {{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}};
    int position;
    char currentPlayer = 'X';
    int turns = 0;

    cout << "Welcome to Tic-Tac-Toe!\n";

    do {
        printBoard(board);
        cout << "Player " << currentPlayer << " turn (choose a position 1-9): ";
        cin >> position;

        if (position < 1 || position > 9) {
            cout << "Invalid input. Choose a position between 1 and 9.\n";
            continue;
        }

        if (!placeMarker(board, position, currentPlayer)) {
            cout << "Position already taken. Choose another position.\n";
            continue;
        }

        if (checkWin(board)) {
            printBoard(board);
            cout << "Player " << currentPlayer << " wins!\n";
            return 0;
        }

        turns++;

        if (turns == 9) {
            printBoard(board);
            cout << "It's a draw!\n";
            return 0;
        }
```

```
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    } while (!checkWin(board));

    return 0;
}

void printBoard(char board[3][3]) {
    cout << "\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (j != 0) cout << "|";
            cout << " " << board[i][j] << " ";
        }
        cout << endl;
        if (i < 2) cout << "-----------\n";
    }
    cout << "\n";
}

bool checkWin(char board[3][3]) {
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] &&
board[i][0] != ' ') return true; // Check rows
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] &&
board[0][i] != ' ') return true; // Check columns
    }
    if ((board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0]
!= ' ') || // Check diagonals
        (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2]
!= ' ')) {
        return true;
    }
    return false;
}

bool placeMarker(char board[3][3], int position, char marker) {
    int row = (position - 1) / 3;
    int col = (position - 1) % 3;

    if (board[row][col] == 'X' || board[row][col] == 'O') {
        return false;
    } else {
        board[row][col] = marker;
        return true;
```

```
    }
}
```

OUTPUT:


Welcome to Tic-Tac-Toe!
| |

| |

| |
Player X turn (choose a position 1-9): 10
Invalid input. Choose a position between 1 and 9.
| |

| |

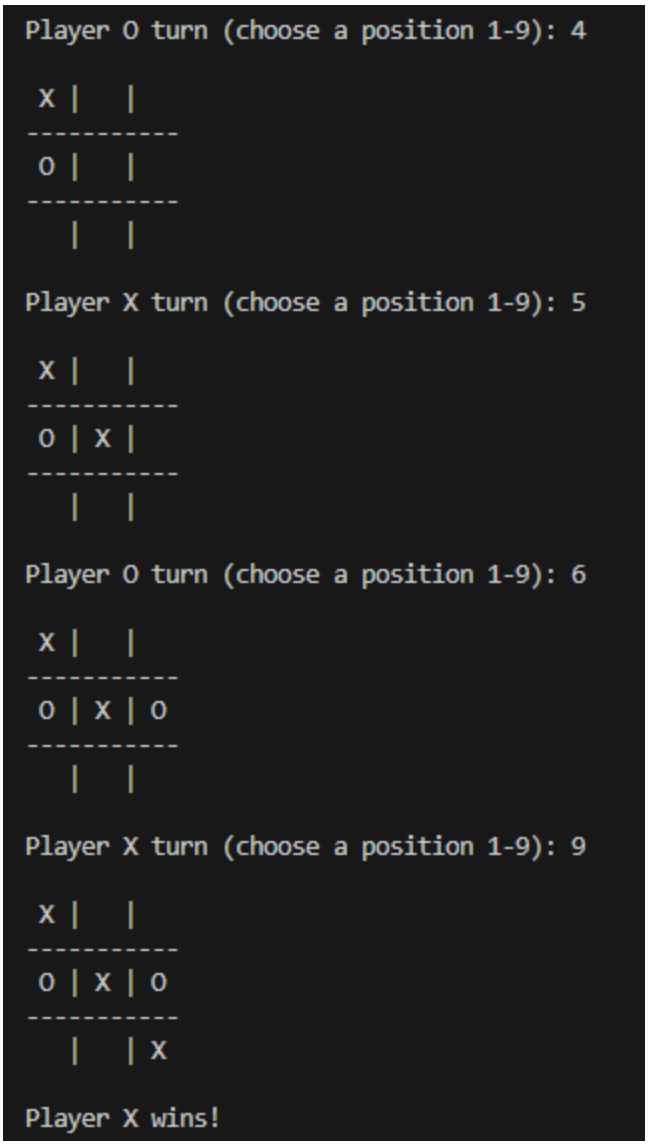
| |
Player X turn (choose a position 1-9): 1
X | |

| |

| |
Player O turn (choose a position 1-9): 1
Position already taken. Choose another position.
X | |

| |

| |



APPROACH:

1. **Initialization:**
   - Initialize a 3x3 game board with empty spaces (' ') to represent an empty Tic-Tac-Toe grid.
   - Initialize the current player as 'X' and set the number of turns to 0.

2. **Game Loop (do-while):**
    - Display the current state of the board.
    - Prompt the current player to choose a position to place their marker ('X' or 'O').
    - Validate the input:
        - Check if the position is within the valid range (1-9).
        - Check if the chosen position is already taken. If it is, ask the player to choose another position.
    - Place the marker on the board at the chosen position.
    - Check for a win:
        - Check if the current player has won by examining rows, columns, and diagonals.
    - Increment in the number of turns.
    - If all positions are filled and no winner is found, declare a draw.
    - Switch the player for the next turn.

3. **Functions:**
    - printBoard(): Displays the current state of the board in a user-friendly manner.
    - checkWin(): Checks if the current player has won by examining rows, columns, and diagonals.
    - place marker (): Places the player's marker on the board at the chosen position if it's not already occupied.

4. **Function Details:**
    - printBoard(): Iterates through the board array and displays the grid with separators for rows and columns.
    - checkWin(); Checks rows, columns, and diagonals for three consecutive markers ('X' or 'O') to determine if there's a win.
    - placeMarker(); Converts the user input position to a board coordinate and places the marker if the chosen position is valid and not already taken.

5. **Termination:**
    - The game loop continues until either a player wins or the board is filled. The game terminates with a victory message, a draw message, or if the players decide to exit.

6. **Input Validation:**
    - The program checks for valid inputs to ensure players enter positions within the range (1-9) and handle invalid or already taken positions.