Problem Statement: Tic Tac Toe Game Development in C

Objective: Design and implement a two-player Tic Tac Toe game using the C programming language. The game should allow players to take turns placing their marks (X or O) on a 3x3 grid, ensuring proper validation of inputs and determining the outcome (win, draw, or ongoing game) after each move.

Requirements:

1. **Game Setup:**
    - The game consists of a 3x3 grid.
    - Two players: Player 1 uses 'X' and Player 2 uses 'O'.
2. **Game Flow:**
    - Initialize an empty board.
    - Alternate turns between Player 1 and Player 2.
    - Prompt the current player to enter their move as a row and column index (1-based).
    - Validate the input to ensure:
        - The chosen cell is within the board range.
        - The cell is not already occupied.
    - Place the player's mark on the board if the move is valid; otherwise, prompt the player to enter a valid move.
3. **Outcome Evaluation:**
    - After each move, check if the current player has won:
        - A win occurs if the player has three of their marks in a row, column, or diagonal.
    - Check if the board is full and declare the game a draw if no player has won.
    - If neither condition is met, continue the game.
4. **Game End:**
    - Announce the winner or declare a draw.
    - Provide an option to restart the game or exit.

Constraints:

1. Input for row and column must be integers within the range 1-3.
2. Handle invalid inputs gracefully (e.g., out-of-range values, non-integer input).
3. Ensure the program runs in a terminal-based interface.

Deliverables:

- A fully functional C program with the following:
    - User-friendly interface to display the board and prompt for player inputs.
    - Clear instructions for gameplay.

- Robust error handling for invalid inputs.
- Logic to evaluate game outcomes (win/draw).

Bonus Challenges:

1. Add a scoreboard to track wins and draws across multiple rounds.
2. Allow one player to play against the computer with a simple AI.
3. Highlight the winning combination on the board.

Expected Output Example:

Welcome to Tic Tac Toe!

```
 1 | 2 | 3

-----------

 4 | 5 | 6

-----------

 7 | 8 | 9
```

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

```
X |   |

-----------

  |   |

-----------

  |   |
```

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

```
X |   |

-----------
```

| O |

| |

... (gameplay continues)

Player 1 wins!

```c
#include <stdio.h>
#define SIZE 3

void initialize(char board[SIZE][SIZE]);
void display(char board[SIZE][SIZE]);
int valid(char board[SIZE][SIZE], int row, int col);
int checkWin(char board[SIZE][SIZE], char mark);
int isDraw(char board[SIZE][SIZE]);
void playGame();

int main() {
    char playAgain;
    do {
        playGame();
        printf("Play again? (y/n): ");
        scanf(" %c", &playAgain);
    } while (playAgain == 'y' || playAgain == 'Y');

    printf("Thanks for playing!\n");
    return 0;
}
void initialize(char board[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++)
        for (int j = 0; j < SIZE; j++)
            board[i][j] = ' ';
}

void display(char board[SIZE][SIZE]) {
    printf("\n");
    for (int i = 0; i < SIZE; i++) {
```

```
        for (int j = 0; j < SIZE; j++) {
            printf(" %c ", board[i][j]);
            if (j < SIZE - 1) printf("|");
        }
        printf("\n");
        if (i < SIZE - 1) printf("---|---|---\n");
    }
    printf("\n");
}

int valid(char board[SIZE][SIZE], int row, int col) {
    return row >= 0 && row < SIZE && col >= 0 && col < SIZE &&
board[row][col] == ' ';
}

int checkWin(char board[SIZE][SIZE], char mark) {
    for (int i = 0; i < SIZE; i++)
        if ((board[i][0] == mark && board[i][1] == mark && board[i][2]
== mark) ||
            (board[0][i] == mark && board[1][i] == mark && board[2][i]
== mark))
            return 1;
    return (board[0][0] == mark && board[1][1] == mark && board[2][2]
== mark) ||
           (board[0][2] == mark && board[1][1] == mark && board[2][0]
== mark);
}

int isDraw(char board[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++)
        for (int j = 0; j < SIZE; j++)
            if (board[i][j] == ' ')
                return 0;
    return 1;
}

void playGame() {
    char board[SIZE][SIZE];
    int row, col, currentPlayer = 1;
    initialize(board);

    while (1) {
        display(board);
```

```
        printf("Player %d (%c), enter row and column (1-3): ",
currentPlayer, currentPlayer == 1 ? 'X' : 'O');
        scanf("%d %d", &row, &col);

        if (!valid(board, row--, col--)) {
            printf("Invalid move, try again.\n");
            continue;
        }

        board[row][col] = (currentPlayer == 1) ? 'X' : 'O';

        if (checkWin(board, board[row][col])) {
            display(board);
            printf("Player %d (%c) wins!\n", currentPlayer,
board[row][col]);
            break;
        }

        if (isDraw(board)) {
            display(board);
            printf("It's a draw!\n");
            break;
        }

        currentPlayer = 3 - currentPlayer;
    }
}
```