Assignment 6

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define SIZE 3

// Function prototypes

void initializeBoard(char board[SIZE][SIZE]);

void printBoard(char board[SIZE][SIZE]);

bool checkWinner(char board[SIZE][SIZE], char symbol);

bool isFull(char board[SIZE][SIZE]);

void getPlayerMove(char board[SIZE][SIZE]);

void getComputerMove(char board[SIZE][SIZE]);

// Main function

int main() {

char board[SIZE][SIZE];

char playerSymbol = 'X';

char computerSymbol = 'O';

char currentTurn = computerSymbol; // Computer starts

bool gameOver = false;

initializeBoard(board);

printf("Welcome to Tic Tac Toe!\n");

printBoard(board);

while (!gameOver) {

if (currentTurn == playerSymbol) {

printf("Player's Turn:\n");

getPlayerMove(board);

} else {

printf("Computer's Turn:\n");

getComputerMove(board);

}

printBoard(board);

// Check for a winner

if (checkWinner(board, currentTurn)) {

printf("%s wins!\n", (currentTurn == playerSymbol) ? "Player" : "Computer");

gameOver = true;

}

// Check for a draw

else if (isFull(board)) {

printf("It's a draw!\n");

gameOver = true;

}

// Switch turns

currentTurn = (currentTurn == playerSymbol) ? computerSymbol : playerSymbol;

}

return 0;

}

// Initialize the board with empty spaces

void initializeBoard(char board[SIZE][SIZE]) {

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

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

board[i][j] = ' ';

}

}

}

// Print the current state of the board

void printBoard(char board[SIZE][SIZE]) {

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");

}

}

// Check if a player has won

bool checkWinner(char board[SIZE][SIZE], char symbol) {

// Check rows and columns

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

if ((board[i][0] == symbol && board[i][1] == symbol && board[i][2] == symbol) ||

(board[0][i] == symbol && board[1][i] == symbol && board[2][i] == symbol)) {

return true;

}

}

// Check diagonals

if ((board[0][0] == symbol && board[1][1] == symbol && board[2][2] == symbol) ||

(board[0][2] == symbol && board[1][1] == symbol && board[2][0] == symbol)) {

return true;

}

return false;

}

// Check if the board is full

bool isFull(char board[SIZE][SIZE]) {

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

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

if (board[i][j] == ' ') {

return false;

}

}

}

return true;

}

// Get the player's move

void getPlayerMove(char board[SIZE][SIZE]) {

int move;

while (1) {

printf("Enter your move (1-9): ");

scanf("%d", &move);

move--; // Convert 1-9 to 0-8

int row = move / SIZE;

int col = move % SIZE;

if (move >= 0 && move < SIZE \* SIZE && board[row][col] == ' ') {

board[row][col] = 'X';

break;

} else {

printf("Invalid move! Try again.\n");

}

}

}

// Get the computer's move

void getComputerMove(char board[SIZE][SIZE]) {

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

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

if (board[i][j] == ' ') {

board[i][j] = 'O';

return;

}

}

}

}

Output:

