# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 10

1) Write a tic-tac-toe program in 3 different languages (Programming Assignment For comparative study of Different Paradigms)

JAVA CODE:

import java.util.Scanner;

public class m {

    private *char*[][] board;

    private *char* currentPlayer;

     m() {

        board = new *char*[3][3];

        currentPlayer = 'X';

        initializeBoard();

    }

    public *void* initializeBoard() {

        for (*int* i = 0; i < 3; i++) {

            for (*int* j = 0; j < 3; j++) {

                board[i][j] = '-';

            }

        }

    }

    public *void* printBoard() {

        for (*int* i = 0; i < 3; i++) {

            for (*int* j = 0; j < 3; j++) {

                System.out.print(board[i][j] + " ");

            }

            System.out.println();

        }

    }

    public *boolean* isBoardFull() {

        for (*int* i = 0; i < 3; i++) {

            for (*int* j = 0; j < 3; j++) {

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

                    return false;

                }

            }

        }

        return true;

    }

    public *boolean* checkForWin() {

        for (*int* i = 0; i < 3; i++) {

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

                return true;

            }

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

                return true;

            }

        }

        if (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) {

            return true;

        }

        if (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer) {

            return true;

        }

        return false;

    }

    public *void* changePlayer() {

        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

    }

    public *void* play() {

*Scanner* scanner = new Scanner(System.in);

        while (true) {

            System.out.println("Current board:");

            printBoard();

            System.out.println("Player " + currentPlayer + ", enter your move (row and column): ");

*int* row = scanner.nextInt();

*int* col = scanner.nextInt();

            if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != '-') {

                System.out.println("This move is not valid");

                continue;

            }

            board[row][col] = currentPlayer;

            if (checkForWin()) {

                System.out.println("Player " + currentPlayer + " wins!");

                printBoard();

                break;

            }

            if (isBoardFull()) {

                System.out.println("The game is a draw!");

                printBoard();

                break;

            }

            changePlayer();

        }

        scanner.close();

    }

    public static *void* main(*String*[] *args*) {

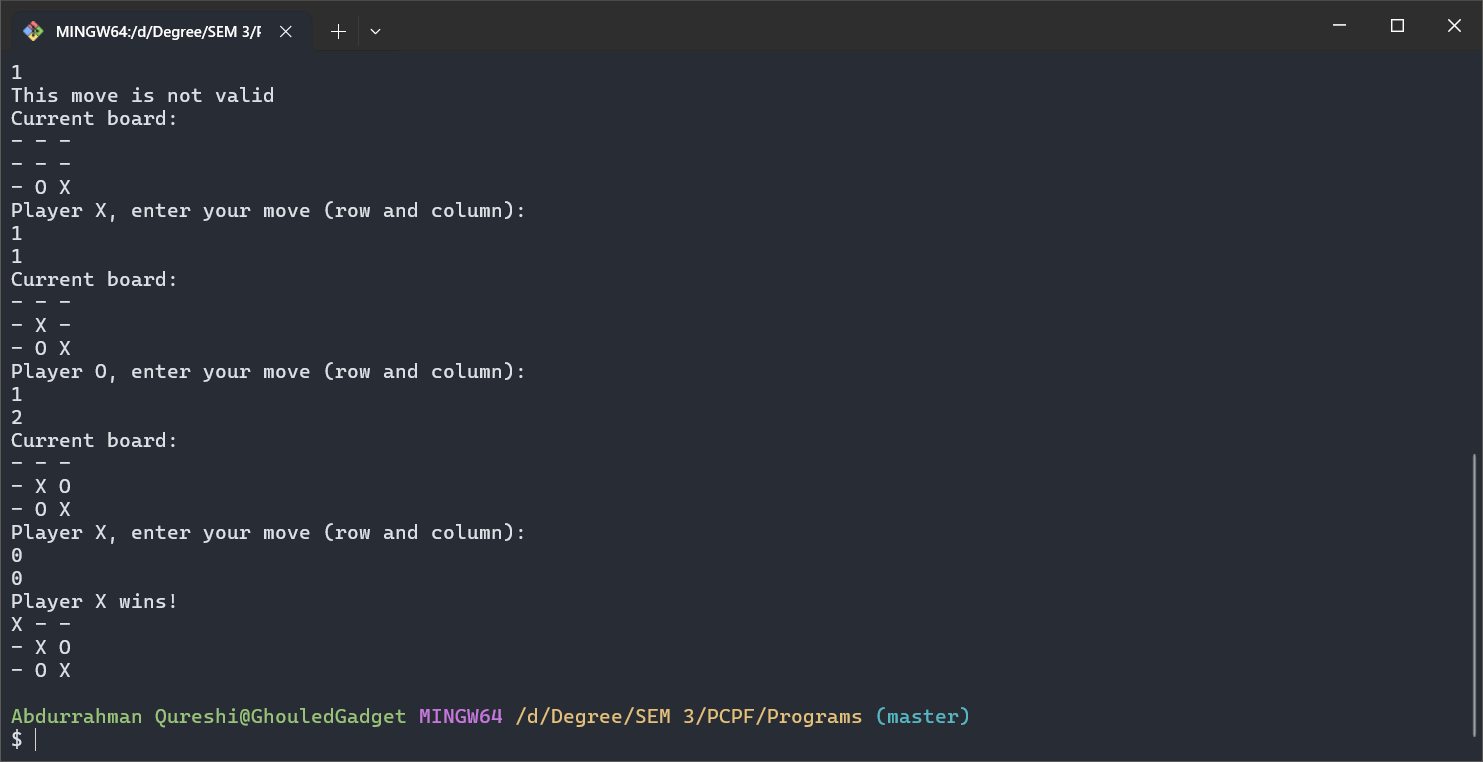
        m game = new m();

        game.play();

    }

}

OUTPUT:



HASKELL CODE:

import Data.List (intersperse, transpose)

import Data.Char (digitToInt)

import Text.Read (readMaybe)

type *Board* = [[*Char*]]

-- Initialize an empty board

emptyBoard :: *Board*

emptyBoard = replicate 3 (replicate 3 ' ')

-- Display the board

printBoard :: *Board* -> *IO* ()

printBoard b = putStrLn $ unlines $ map (intersperse '|') b

-- Check if a player has won

checkWin :: *Board* -> *Char* -> *Bool*

checkWin board player =

    let rows = board

        cols = transpose board

        diag1 = [board !! i !! i | i <- [0..2]]

        diag2 = [board !! i !! (2 - i) | i <- [0..2]]

    in any (all (== player)) (rows ++ cols ++ [diag1, diag2])

-- Check if the board is full

isFull :: *Board* -> *Bool*

isFull = all (all (/= ' '))

-- Make a move on the board

makeMove :: *Board* -> *Int* -> *Int* -> *Char* -> *Board*

makeMove board row col player =

    take row board ++ [take col (board !! row) ++ [player] ++ drop (col + 1) (board !! row)] ++ drop (row + 1) board

-- Function to parse user input

parseInput :: *String* -> *Maybe* (*Int*, *Int*)

parseInput input =

    case words input of

        [r, c] -> do

            row <- readMaybe r

            col <- readMaybe c

            if row >= 0 && row <= 2 && col >= 0 && col <= 2

                then return (row, col)

                else Nothing

        \_ -> Nothing

-- Main function to play the game

main :: *IO* ()

main = playGame emptyBoard 'X'

-- Recursive function to continue the game

playGame :: *Board* -> *Char* -> *IO* ()

playGame board player = do

    printBoard board

    if checkWin board (if player == 'X' then 'O' else 'X')

        then putStrLn $ "Player " ++ [if player == 'X' then 'O' else 'X'] ++ " wins!"

        else if isFull board

            then putStrLn "It's a draw!"

            else do

                putStrLn $ "Player " ++ [player] ++ ", enter row and column (0-2) separated by a space (e.g., '1 2'):"

                input <- getLine

                case parseInput input of

                    Just (row, col) ->

                        if board !! row !! col /= ' '

                            then putStrLn "Invalid move, try again." >> playGame board player

                            else playGame (makeMove board row col player) (if player == 'X' then 'O' else 'X')

                    Nothing -> putStrLn "Invalid input, try again." >> playGame board player

OUTPUT:



C++ CODE:

#include <stdio.h>

*char* board[3][3];

*char* currentPlayer = 'X';

*void* initBoard() {

    for (*int* i = 0; i < 3; i++)

        for (*int* j = 0; j < 3; j++)

            board[i][j] = ' ';

}

*void* printBoard() {

    printf(" %c | %c | %c\n", board[0][0], board[0][1], board[0][2]);

    printf("---|---|---\n");

    printf(" %c | %c | %c\n", board[1][0], board[1][1], board[1][2]);

    printf("---|---|---\n");

    printf(" %c | %c | %c\n", board[2][0], board[2][1], board[2][2]);

}

*int* checkWin() {

    for (*int* i = 0; i < 3; i++) {

        if (board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer) return 1;

        if (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer) return 1;

    }

    if (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) return 1;

    if (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer) return 1;

    return 0;

}

*void* switchPlayer() {

    currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

}

*int* isBoardFull() {

    for (*int* i = 0; i < 3; i++)

        for (*int* j = 0; j < 3; j++)

            if (board[i][j] == ' ') return 0;

    return 1;

}

*int* main() {

*int* row, col;

    initBoard();

    while (1) {

        printBoard();

        printf("Player %c, enter row and column (0-2): ", currentPlayer);

        scanf("%d %d", &row, &col);

        if (row < 0 || row > 2 || col < 0 || col > 2 || board[row][col] != ' ') {

            printf("Invalid move, try again.\n");

            continue;

        }

        board[row][col] = currentPlayer;

        if (checkWin()) {

            printBoard();

            printf("Player %c wins!\n", currentPlayer);

            break;

        }

        if (isBoardFull()) {

            printBoard();

            printf("It's a draw!\n");

            break;

        }

        switchPlayer();

    }

    return 0;

}

OUTPUT:

