24th September, 2024 Tuesday — Laboratory 1

1) Implement the Tic-Tac-Toe Game, after having written an algorithm for the same.

Algorithm for the Tic-Tac-Toe Game

Step 1: Initialize the Game Board:
Create a 3×3 matrix (two-dimensional array) with empty values

Step 2: Display Board Function
Function prints the current state of the board.

Step 3: User Input Function
Function allows the user to input their move.

Step 4: Win Checker Function
Function to check if a player has won the game

Step 5: Draw Checker Function
Function to check if there is a condition for draw — the board if full and there can be no winner.

Step 6: Algorithm for the Computer's Move:

Conditions for draw.

X	O	O
X	O	X
O	X	O

Similarly.

Step 7: Main Game Loop:
while the game is going on.
{ Step (2); Step (3); Step (4), Step (5), Step (6), Step (5) }

Source code (in python):

```
import random
board = [[' ', ' ', ' '], [' ', ' ', ' '], [' ', ' ', ' ']]

def print_board():
    for row in board:
        print(" | ".join(cell if cell != ' ' else ' ' for cell in row))
        print("-" * 9)


def start():
    print("Welcome to tic-tac-toe!")
    random_num = random.randint(0, 1)
    if random_num == 0:
        print("Player plays first.")
    else:
        print("Computer plays first.")
        computer_plays()

while True:
    if random_num == 0:
        player_plays()
        if check_win('X'):
            print_board()
            print("Player won!")
            return
        random_num = 1
    else:
        computer_plays()
        if check_win('O'):
            print_board()
            print("Computer won!")
            return
        random_num = 0
```

```
if board_full():
    print_board()
    print("Draw!")
    return

def check_win(player):
    wins = [
        [(0,0), (0,1), (0,2)],
        [(1,0), (1,1), (1,2)],
        [(2,0), (2,1), (2,2)],
        [(0,0), (1,0), (2,0)],
        [(0,1), (1,1), (2,1)],
        [(0,2), (1,2), (2,2)],
        [(0,0), (1,1), (2,2)],
        [(0,2), (1,1), (2,0)],
    ]

    for win in wins:
        if all(board[x][y] == player for x,y in win):
            return True
    return False

def player_plays():
    print_board()
    while True:
        try:
            a,b = map(int, input("Enter the co-ordinates (row and
                column 0-2, eg. '0 1'): ").split())
            if board[a][b] == ' ':
                board[a][b] = 'X'
                break
            else:
                print("Cell already taken. Try again")
        except (ValueError, IndexError):
```

print("Invalid Input. Please enter co-ordinates in the format row-column!")

```
def computer_plays():
    _, move = algo(board, '0')
    board[move[0]][move[1]] = '0'


def algo(board, current_player):
    if check_win('X'):
        return -1, None
    elif check_win('0'):
        return 1, None
    elif board_full():
        return 0, None
    if current_player == '0':
        best_score = -float('inf')
        best_move = None
        for i in range(3):
            for j in raye(3):
                if board[i][j] == '':
                    board[i][j] = '0'
                    score, _ = algo(board, 'X')
                    board[i][j] = ''
                    if score > best_score:
                        best_score = score
                        best_move = (i, j)
        return best_score, best_move
    else:
        best_score = float('inf')
        best_move = None
        for i in range(3):
            for j in range(3):
                if board[i][j] == '':
                    board[i][j] = 'X'
```

```
score, _ = algo(board, '0')
board[i][j] = ''
if score < best_score:
    best_score = score
    best_mov = (i, j)
return best_score, best_mov

def board_full():
    return all(all(i != '' for i in row) for row in board)

start()
```