Lab 1

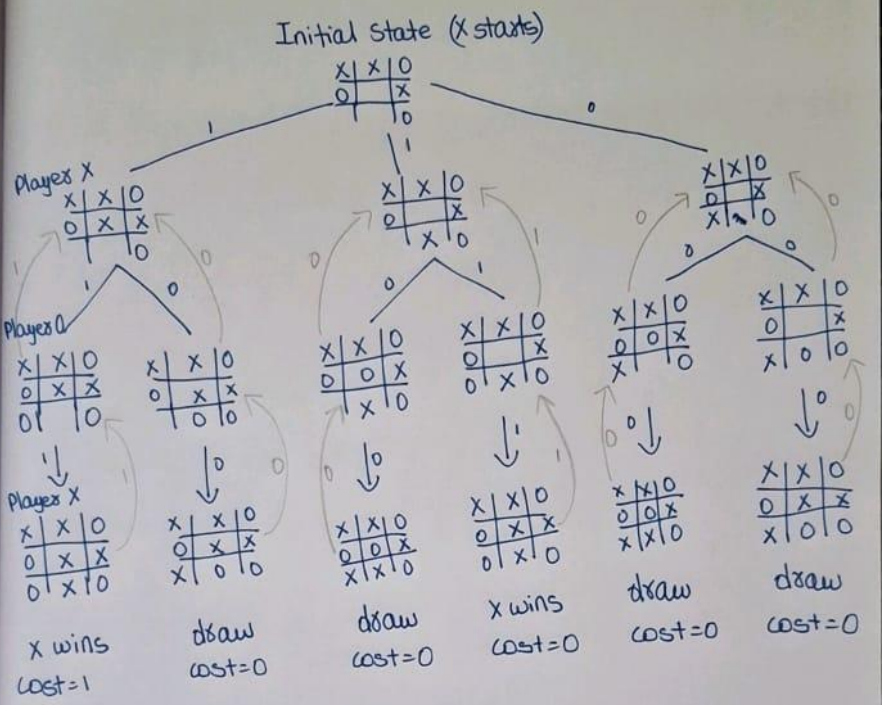
Implement Tic –Tac –Toe Game.

Algorithm:

18/8/25

LAB- I

Implement tic-tac-toe game.



Algorithm:

Step 1: Start

Step 2: Initialize 3x3 grid with empty spaces

Step 3: Initialize current Player to X

Step 4: Display initial state

Step 5: Input row & column values

Step 6: Check if position is valid & empty

Step 7: If invalid go back to step 5

Step 8: If valid place marker in that position

Step 9: Check if current player wins

Step 10: If yes display Current player wins and go to step 13

Step 11: If ~~draw~~ no empty spaces & nobody wins display draw and go to step 13

Step 12: Switch player

Step 13: Stop

18/08

Code:

```
def print_board(board):
    for row in board:
        print(" ".join(row))
    print()

def check_winner(board, player):
    for i in range(3):
        if all(board[i][j] == player for j in range(3)):
            return True
        if all(board[j][i] == player for j in range(3)):
            return True
    if all(board[i][i] == player for i in range(3)):
        return True
    if all(board[i][2 - i] == player for i in range(3)):
        return True
    return False

def is_draw(board):
    return all(board[i][j] != '-' for i in range(3) for j in range(3))

cost_counter = 0

def minimax(board, is_ai_turn):
    global cost_counter
    cost_counter += 1
    if check_winner(board, 'O'):
```

```
    return 1
if check_winner(board, 'X'):
    return -1
if is_draw(board):
    return 0

if is_ai_turn:
    best_score = -float('inf')
    for i in range(3):
        for j in range(3):
            if board[i][j] == '-':
                board[i][j] = 'O'
                score = minimax(board, False)
                board[i][j] = '-'
                best_score = max(score, best_score)
    return best_score
else:
    best_score = float('inf')
    for i in range(3):
        for j in range(3):
            if board[i][j] == '-':
                board[i][j] = 'X'
                score = minimax(board, True)
                board[i][j] = '-'
                best_score = min(score, best_score)
    return best_score
```

```
def manual_game():
    board = [['-' for _ in range(3)] for _ in range(3)]
    print("Initial Board:")
    print_board(board)

    while True:

        while True:
            try:
                x_row = int(input("Enter X row (1-3): ")) - 1
                x_col = int(input("Enter X col (1-3): ")) - 1
                if board[x_row][x_col] == '-':
                    board[x_row][x_col] = 'X'
                    break
                else:
                    print("Cell occupied!")
            except:
                print("Invalid input!")

        print("Board after X move:")
        print_board(board)

        if check_winner(board, 'X'):
            print("X wins!")
            break
        if is_draw(board):
            print("Draw!")
```

```
    break

while True:
    try:
        o_row = int(input("Enter O row (1-3): ")) - 1
        o_col = int(input("Enter O col (1-3): ")) - 1
        if board[o_row][o_col] == '-':
            board[o_row][o_col] = 'O'
            break
        else:
            print("Cell occupied!")
    except:
        print("Invalid input!")

print("Board after O move:")
print_board(board)

if check_winner(board, 'O'):
    print("O wins!")
    break
if is_draw(board):
    print("Draw!")
    break
global cost_counter
cost_counter = 0
cost = minimax(board, True)
print(f"AI evaluation cost from this position: {cost_counter} states examined")
```

```
        print(f"AI evaluation score from this position: {cost}")

manual_game()
```

Output: