Name: Vatsa Nagaria
UID: 2019130041
TE COMPS
Batch C
Date: 14/11/21

EXPERIMENT 3

Aim: To implement the game, Tic-Tac-Toe using the A* search strategy.

Code:

```
import random
import time


n = [0, 1, 2, 3, 4, 5, 6, 7, 8]
board = ['0', '0', '0', '0', '0', '0', '0', '0', '0']
rows = [['0', '0', '0'], ['0', '0', '0'], ['0', '0', '0']]
columns = [['0', '0', '0'], ['0', '0', '0'], ['0', '0', '0']]
diagonals = [['0', '0', '0'], ['0', '0', '0']]


def player_win(player):
    over = False
    for i in range(3):
        if rows[i].count(player) == 3:
            over = True
            break

        if columns[i].count(player) == 3:
            over = True
            break

        if i < 2:
            if diagonals[i].count(player) == 3:
                over = True
                break
```

```
    if over:
        return True
    else:
        return False


def check():
    win_X = []
    win_O = []
    for i in range(3):
        if rows[i].count('0') == 1:
            index_zero = rows[i].index('0')
            index_zero = 3 * i + index_zero
            if rows[i].count('X') == 2:
                win_X.append(index_zero)
            elif rows[i].count('O') == 2:
                win_O.append(index_zero)

        if columns[i].count('0') == 1:
            index_zero = columns[i].index('0')
            index_zero = i + (index_zero * 3)
            if columns[i].count('X') == 2:
                win_X.append(index_zero)
            elif columns[i].count('O') == 2:
                win_O.append(index_zero)
        if i < 2:
            if diagonals[i].count('0') == 1:
                index_zero = diagonals[i].index('0')
                if i == 0:
                    index_zero = 4 * index_zero
                elif i == 1:
                    index_zero = 2 + (index_zero * 2)
                if diagonals[i].count('X') == 2:
                    win_X.append(index_zero)
                elif diagonals[i].count('O') == 2:
                    win_O.append(index_zero)

    return [win_O, win_X]
```

```
def cal_heuristic(turn):
    available = []
    for i in range(9):
        if board[i] == '0':
            available.append(i)

    heuristic_values = []
    heuristic_values_choices = []

    for possible in available:
        X_combo = 0
        O_combo = 0

        x = possible // 3
        y = possible % 3
        board[possible] = turn
        rows[x][y] = turn
        columns[y][x] = turn
        if x == y:
            diagonals[0][x] = turn
            if x == 1:
                diagonals[1][x] = turn
        if possible == 2 or possible == 6:
            z = possible // 2 - 1
            diagonals[1][z] = turn

        empty_minus_one = len(available) - 1

        for i in range(3):
            if rows[i].count('0') == 3:
                if (empty_minus_one == 5):
                    if turn == 'O':
                        X_combo = X_combo + 1
                    else:
                        O_combo = O_combo + 1
            elif rows[i].count('0') == 2:
                if empty_minus_one > 3:
                    if rows[i].count('X') == 1:
                        X_combo = X_combo + 1
```

```
            elif rows[i].count('O') == 1:
                O_combo = O_combo + 1
        elif empty_minus_one == 3:
            if rows[i].count('X') == 1:
                if turn == 'O':
                    X_combo = X_combo + 1
            elif rows[i].count('O') == 1:
                if turn == 'X':
                    O_combo = O_combo + 1
    elif rows[i].count('0') == 1:
        if empty_minus_one > 1:
            if rows[i].count('X') == 2:
                X_combo = X_combo + 1
            elif rows[i].count('O') == 2:
                O_combo = O_combo + 1
        elif empty_minus_one == 1:
            if rows[i].count('X') == 2:
                if turn == 'O':
                    X_combo = X_combo + 1
            elif rows[i].count('O') == 2:
                if turn == 'X':
                    O_combo = O_combo + 1

    if columns[i].count('0') == 3:
        if (empty_minus_one == 5):
            if turn == 'O':
                X_combo = X_combo + 1
            else:
                O_combo = O_combo + 1
    elif columns[i].count('0') == 2:
        if empty_minus_one > 3:
            if columns[i].count('X') == 1:
                X_combo = X_combo + 1
            else:
                O_combo = O_combo + 1
        elif empty_minus_one == 3:
            if columns[i].count('X') == 1:
                if turn == 'O':
                    X_combo = X_combo + 1
            elif columns[i].count('O') == 1:
```

```
            if turn == 'X':
                O_combo = O_combo + 1
    elif columns[i].count('0') == 1:
        if empty_minus_one > 1:
            if columns[i].count('X') == 2:
                X_combo = X_combo + 1
            elif columns[i].count('O') == 2:
                O_combo = O_combo + 1
        elif empty_minus_one == 1:
            if columns[i].count('X') == 2:
                if turn == 'O':
                    X_combo = X_combo + 1
            elif columns[i].count('O') == 2:
                if turn == 'X':
                    O_combo = O_combo + 1

    if i < 2:
        if diagonals[i].count('0') == 3:
            if (empty_minus_one == 5):
                if turn == 'O':
                    X_combo = X_combo + 1
                else:
                    O_combo = O_combo + 1
        elif diagonals[i].count('0') == 2:
            if empty_minus_one > 3:
                if diagonals[i].count('X') == 1:
                    X_combo = X_combo + 1
                else:
                    O_combo = O_combo + 1
            elif empty_minus_one == 3:
                if diagonals[i].count('X') == 1:
                    if turn == 'O':
                        X_combo = X_combo + 1
                elif diagonals[i].count('O') == 1:
                    if turn == 'X':
                        O_combo = O_combo + 1
        elif diagonals[i].count('0') == 1:
            if empty_minus_one > 1:
                if diagonals[i].count('X') == 2:
                    X_combo = X_combo + 1
```

```
                elif columns[i].count('O') == 2:
                    O_combo = O_combo + 1
            elif empty_minus_one == 1:
                if diagonals[i].count('X') == 2:
                    if turn == 'O':
                        X_combo = X_combo + 1
                elif diagonals[i].count('O') == 2:
                    if turn == 'X':
                        O_combo = O_combo + 1
    board[possible] = '0'
    rows[x][y] = '0'
    columns[y][x] = '0'
    if x == y:
        diagonals[0][x] = '0'
        if x == 1:
            diagonals[1][x] = '0'
    if possible == 2 or possible == 6:
        z = possible // 2 - 1
        diagonals[1][z] = '0'

    if turn == 'X':
        heuristic_values.append(X_combo - O_combo)
    else:
        heuristic_values.append(O_combo - X_combo)

    heuristic_values_choices.append(possible)

final_heuristics = []
max_value = max(heuristic_values)
for i in range(len(heuristic_values)):
    if heuristic_values[i] == max_value:
        final_heuristics.append(heuristic_values_choices[i])

if len(available) >= 7:
    return random.choice(final_heuristics)

final_positions = []
min_values = []

for choice in final_heuristics:
```

```
            x = choice // 3
            y = choice % 3
            board[choice] = turn
            rows[x][y] = turn
            columns[y][x] = turn
            if x == y:
                diagonals[0][x] = turn
                if x == 1:
                    diagonals[1][x] = turn
            if choice == 2 or choice == 6:
                z = choice // 2 - 1
                diagonals[1][z] = turn

            checks = check()
            if turn == 'X':
                min_values.append(len(checks[0]))
            else:
                min_values.append(len(checks[1]))

            board[choice] = '0'
            rows[x][y] = '0'
            columns[y][x] = '0'
            if x == y:
                diagonals[0][x] = '0'
                if x == 1:
                    diagonals[1][x] = '0'
            if choice == 2 or choice == 6:
                z = choice // 2 - 1
                diagonals[1][z] = '0'

        min_val = min(min_values)
        for i in range(len(min_values)):
            if min_values[i] == min_val:
                final_positions.append(final_heuristics[i])

        return random.choice(final_positions)


    def main():
        game_won = False
```

```
moves = 0
turn = 'X'
print("You are " + turn + " and Computer is O")

while moves < 9:
    for i in range(3):
        for j in range(3):
            if rows[i][j] == '0':
                print('_', end=' ')
            else:
                print(rows[i][j], end=" ")
        print()
    print()

    if moves % 2 == 0:
        choice = int(input("Enter your choice: "))
        while choice not in n:
            print('Please choose from 0-8')
            choice = int(input("Enter your choice: "))

    else:
        if moves < 2:
            choice = random.choice(n)
        else:
            my_win = check()
            choice = -1
            if turn == 'O':
                if len(my_win[0]) > 0:
                    choice = my_win[0][0]
            elif turn == 'X':
                if len(my_win[1]) > 0:
                    choice = my_win[1][0]
            if choice == -1:
                if turn == 'O':
                    if len(my_win[1]) > 0:
                        choice = my_win[1][0]
                elif turn == 'X':
                    if len(my_win[0]) > 0:
                        choice = my_win[0][0]
```

```
            if choice == -1:
                choice = cal_heuristic(turn)

        n.remove(choice)
        if turn == 'O':
            print("Computer placed O at " + str(choice))
        x = choice // 3
        y = choice % 3
        board[choice] = turn
        rows[x][y] = turn
        columns[y][x] = turn
        if x == y:
            diagonals[0][x] = turn
            if x == 1:
                diagonals[1][x] = turn
        if choice == 2 or choice == 6:
            z = choice // 2 - 1
            diagonals[1][z] = turn

        moves = moves + 1

        if player_win(turn):
            if turn == 'X':
                print("\n**Congratulations**\n YOU WON!!\n")
            else:
                print("\nYOU LOST!!\n")
            game_won = True
            break

        if turn == 'X':
            turn = 'O'
        else:
            turn = 'X'

        time.sleep(0.5)

    for i in range(3):
        for j in range(3):
            if rows[i][j] == '0':
                print('_', end=' ')
```

```
            else:
                print(rows[i][j], end=" ")
        print()
    print()

    if not game_won:
        print("\nITS A TIE!!")


if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\vatsa\PycharmProjects\exp3\venv\Scripts\python.exe C:/Users/vatsa/PycharmProjects/exp3/main.py
You are X and Computer is O
_ _ _
_ _ _
_ _ _

Enter your choice: 0
X _ _
_ _ _
_ _ _

Computer placed O at 6
X _ _
_ _ _
O _ _

Enter your choice: 8
X _ _
_ _ _
O _ X

Computer placed O at 4
X _ _
_ O _
O _ X

Enter your choice: 2
X _ X
_ O _
O _ X
```

```
Computer placed O at 1
X O X
_ O _
O _ X

Enter your choice: 5

**Congratulations**
 YOU WON!!

X O X
_ O X
O _ X


Process finished with exit code 0
```