BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India

DEPARTMENT OF COMPUTER ENGINEERING
SUBJECT: Artificial Intelligence and Machine Learning

Name	Manish Shashikant Jadhav
UID no.	2023301005

Experiment 4

AIM :

Implement the problem using the Informed searching technique min-max algorithm . Analyze the algorithm with respect to Completeness, Optimality, time and space Complexity

a) Tic Tac Toe

CODE:

```
import math



def print_board(board):

    for i in range(3):

        print(" | ".join(board[i*3:(i+1)*3]))

        if i < 2:

            print("---------")



def empty_cells(board):

    return [i for i, cell in enumerate(board) if cell == " "]



def is_winner(board, player):

    winning_combinations = [

        [0, 1, 2], [3, 4, 5], [6, 7, 8],  # Rows

        [0, 3, 6], [1, 4, 7], [2, 5, 8],  # Columns
        [0, 4, 8], [2, 4, 6]  # Diagonals
    ]
    return any(all(board[i] == player for i in combo) for combo in winning_combinations)


def game_over(board):
    return is_winner(board, "X") or is_winner(board, "O") or len(empty_cells(board)) == 0


def minimax(board, depth, is_maximizing):
    if is_winner(board, "X"):
        return -1
    if is_winner(board, "O"):
        return 1
    if len(empty_cells(board)) == 0:
        return 0

    if is_maximizing:
        best_score = -math.inf
        for move in empty_cells(board):
            board[move] = "O"
            score = minimax(board, depth + 1, False)
            board[move] = " "
            best_score = max(score, best_score)

        return best_score

    else:

        best_score = math.inf

        for move in empty_cells(board):

            board[move] = "X"

            score = minimax(board, depth + 1, True)

            board[move] = " "

            best_score = min(score, best_score)

        return best_score


def get_best_move(board):

    best_score = -math.inf

    best_move = None

    for move in empty_cells(board):

        board[move] = "O"

        score = minimax(board, 0, False)

        board[move] = " "

        if score > best_score:

            best_score = score

            best_move = move

    return best_move
```

```
def get_player_move(board, player):
    while True:
        try:
            move = int(input(f"Player {player}, enter your move (0-8): "))
            if move not in empty_cells(board):
                raise ValueError
            return move
        except ValueError:
            print("Invalid move. Try again.")


def play_game(mode):
    board = [" " for _ in range(9)]
    current_player = "X"

    if mode == "1":
        print("You are X, AI is O")
    else:
        print("Player 1: X, Player 2: O")

    print_board(board)
```

```
while not game_over(board):
    if mode == "1" and current_player == "O":
        print("AI is making a move...")
        move = get_best_move(board)
    else:
        move = get_player_move(board, current_player)


    board[move] = current_player
    print_board(board)


    if game_over(board):
        break


    current_player = "O" if current_player == "X" else "X"


if is_winner(board, "X"):
    print("X wins!")
elif is_winner(board, "O"):
    print("O wins!")
else:
    print("It's a tie!")
```

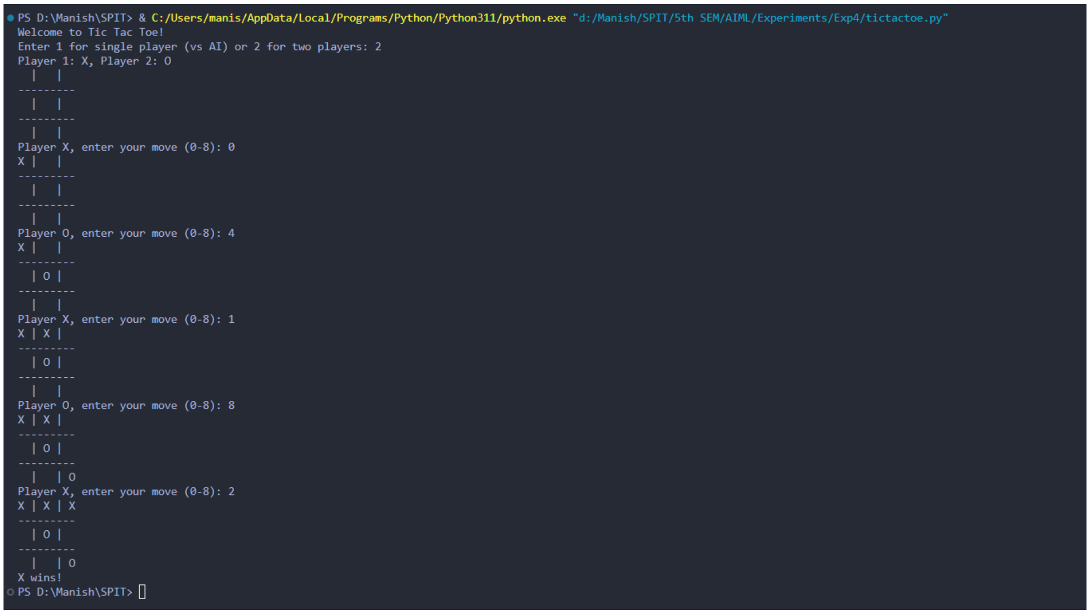
```
def main():
    print("Welcome to Tic Tac Toe!")
    while True:
        mode = input("Enter 1 for single player (vs AI) or 2 for two players: ")
        if mode in ["1", "2"]:
            break
        print("Invalid input. Please enter 1 or 2.")

    play_game(mode)

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

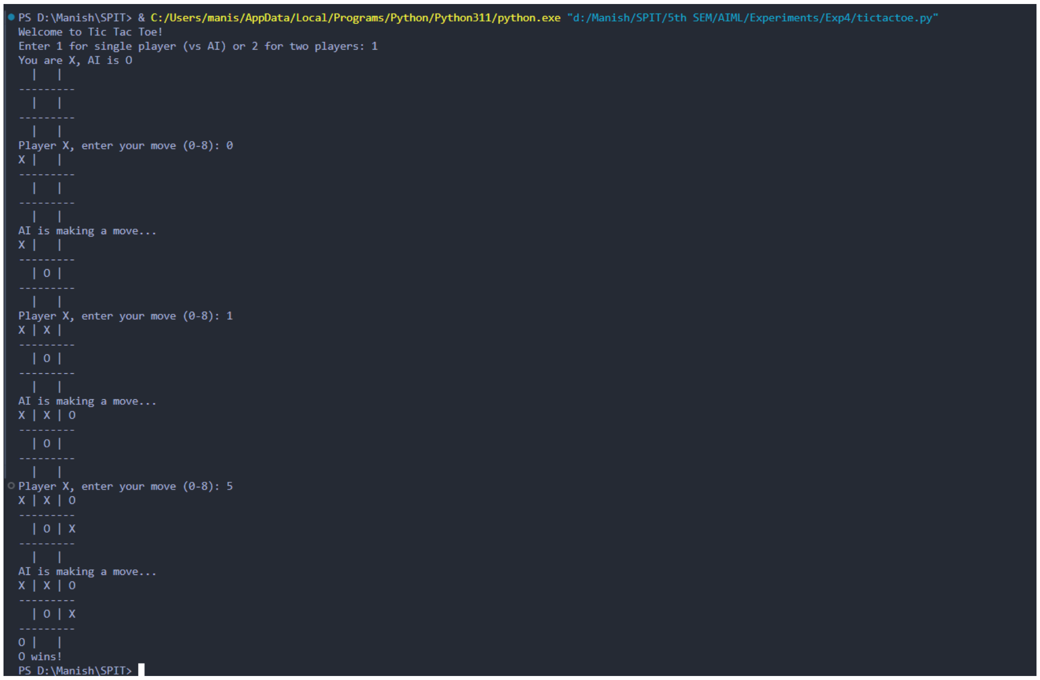
OUTPUT:

1. Two Player Game:



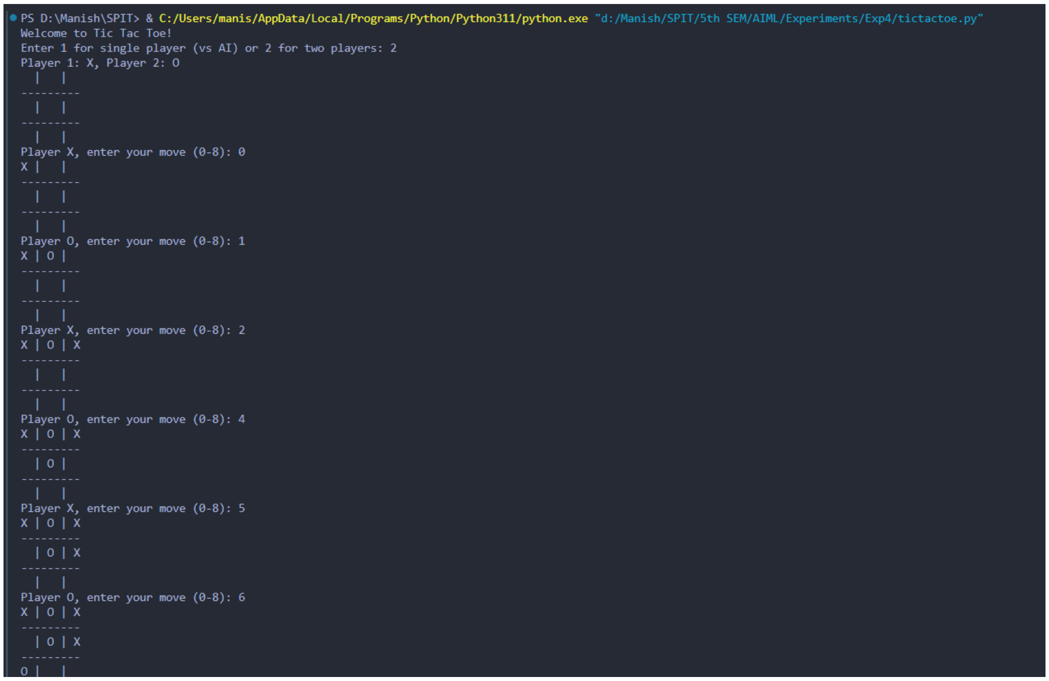
```
● PS D:\Manish\SPIT> & C:/Users/manis/AppData/Local/Programs/Python/Python311/python.exe "d:/Manish/SPIT/5th SEM/AIML/Experiments/Exp4/tictactoe.py"
  Welcome to Tic Tac Toe!
  Enter 1 for single player (vs AI) or 2 for two players: 2
  Player 1: X, Player 2: O
    |   |
  ---------
    |   |
  ---------
    |   |
  Player X, enter your move (0-8): 0
  X |   |
  ---------
    |   |
  ---------
    |   |
  Player O, enter your move (0-8): 4
  X |   |
  ---------
    | O |
  ---------
    |   |
  Player X, enter your move (0-8): 1
  X | X |
  ---------
    | O |
  ---------
    |   |
  Player O, enter your move (0-8): 8
  X | X |
  ---------
    | O |
  ---------
    |   | O
  Player X, enter your move (0-8): 2
  X | X | X
  ---------
    | O |
  ---------
    |   | O
  X wins!
○ PS D:\Manish\SPIT> ▯
```

2. Single Player Game AI Win:

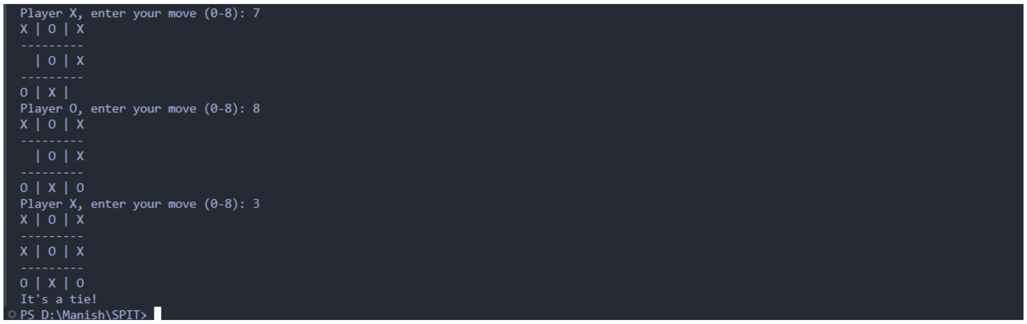


```
PS D:\Manish\SPIT> & C:/Users/manis/AppData/Local/Programs/Python/Python311/python.exe "d:/Manish/SPIT/5th SEM/AIML/Experiments/Exp4/tictactoe.py"
Welcome to Tic Tac Toe!
Enter 1 for single player (vs AI) or 2 for two players: 1
You are X, AI is O
  |   |
---------
  |   |
---------
  |   |
Player X, enter your move (0-8): 0
X |   |
---------
  |   |
---------
  |   |
AI is making a move...
X |   |
---------
  | O |
---------
  |   |
Player X, enter your move (0-8): 1
X | X |
---------
  | O |
---------
  |   |
AI is making a move...
X | X | O
---------
  | O |
---------
  |   |
Player X, enter your move (0-8): 5
X | X | O
---------
  | O | X
---------
  |   |
AI is making a move...
X | X | O
---------
  | O | X
---------
O |   |
O wins!
PS D:\Manish\SPIT>
```

3. Game Tie:



```
PS D:\Manish\SPIT> & C:/Users/manis/AppData/Local/Programs/Python/Python311/python.exe "d:/Manish/SPIT/5th SEM/AIML/Experiments/Exp4/tictactoe.py"
Welcome to Tic Tac Toe!
Enter 1 for single player (vs AI) or 2 for two players: 2
Player 1: X, Player 2: O
  |   |
---------
  |   |
---------
  |   |
Player X, enter your move (0-8): 0
X |   |
---------
  |   |
---------
  |   |
Player O, enter your move (0-8): 1
X | O |
---------
  |   |
---------
  |   |
Player X, enter your move (0-8): 2
X | O | X
---------
  |   |
---------
  |   |
Player O, enter your move (0-8): 4
X | O | X
---------
  | O |
---------
  |   |
Player X, enter your move (0-8): 5
X | O | X
---------
  | O | X
---------
  |   |
Player O, enter your move (0-8): 6
X | O | X
---------
  | O | X
---------
O |   |
```



```
Player X, enter your move (0-8): 7
X | O | X
---------
  | O | X
---------
O | X |
Player O, enter your move (0-8): 8
X | O | X
---------
  | O | X
---------
O | X | O
Player X, enter your move (0-8): 3
X | O | X
---------
X | O | X
---------
O | X | O
It's a tie!
PS D:\Manish\SPIT>
```

Analysis of Algorithm	1. **Completeness:** Yes, the algorithm is complete and will always find a solution (win, lose, or draw). 2. **Optimality:** Yes, the algorithm is optimal for both players when they play optimally. 3. **Time Complexity:** O(b^d), which is O(9!) for Tic-Tac-Toe, equivalent to O(362,880) in the worst case. 4. **Space Complexity:** O(d), which is O(9) for Tic-Tac-Toe, meaning the space complexity is constant and manageable.
CONCLUSION:	Hence by completing this experiment I came to know about Informed searching technique min-max algorithm .