Xavier Institute of Engineering
Mahim, Mumbai 400016
Department of Computer Engineering
(Affiliated to University of Mumbai)

Name	Philip Mathew	Subject	AI Lab
XIE ID	201901030	Experiment No	05
DATE	April 2, 2022	Assigned Date	March 28, 2022

Aim: To implement a game playing algorithm

Theory:

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Code:

```
#initialize the board
board = {1: ' ', 2: ' ', 3: ' ',
         4: ' ', 5: ' ', 6: ' ',
         7: ' ', 8: ' ', 9: ' '}

def printBoard(board):
    print(board[1] + '|' + board[2] + '|' +
board[3])
    print('-+-+-')
    print(board[4] + '|' + board[5] + '|' +
board[6])
    print('-+-+-')
    print(board[7] + '|' + board[8] + '|' +
board[9])
    print("\n")

#printBoard(board)

# if the space is empty we can input 'X' or 'O'
def spaceIsFree(position):
    if board[position] == ' ':
        return True
```

```
    else:
        return False

#print(spaceIsFree(1)) -> returns true if position
1 is free

def insertLetter(letter, position):
    if spaceIsFree(position):
        board[position] = letter
        printBoard(board)
        if (checkDraw()):
            print("Draw!")
            exit()
        if checkForWin():
            if letter == 'X':
                print("Bot wins!")
                exit()
            else:
                print("Player wins!")
                exit()

        return


    else:
        print("Can't insert there!")
        position = int(input("Please enter new
position:  "))
        insertLetter(letter, position)
        return

#insertLetter(x, 1)

def checkDraw():
    for key in board.keys():
```

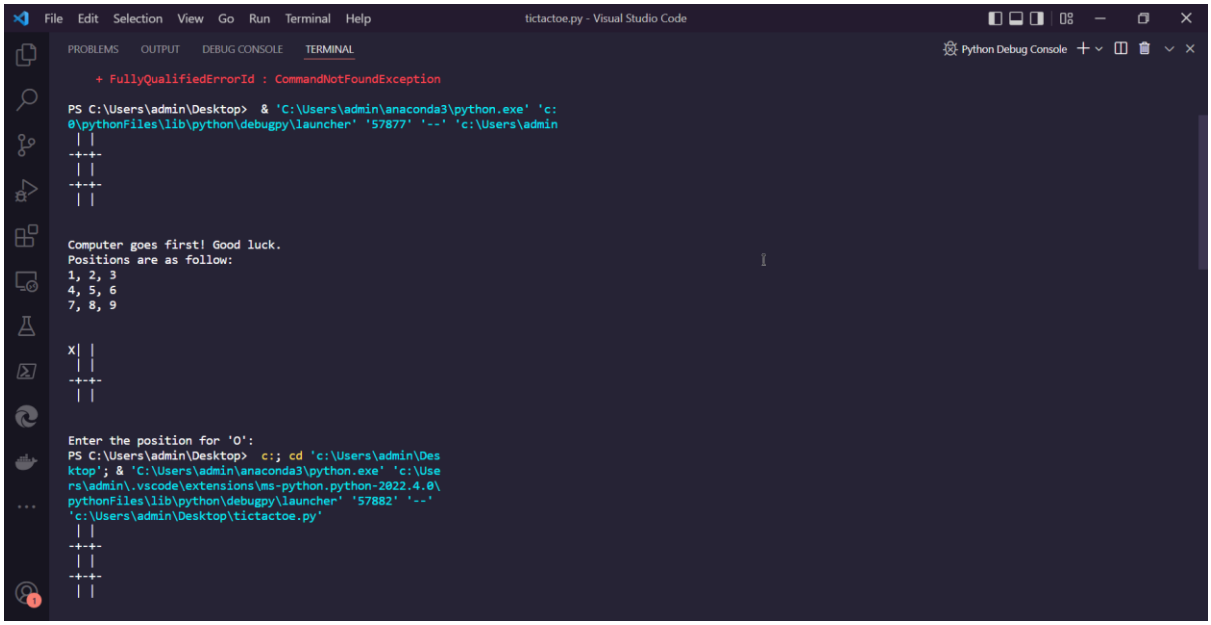
```
        if (board[key] == ' '): #if there are empty
spaces we can still play & not a draw
            return False
    return


def checkForWin():
    if (board[1] == board[2] and board[1] ==
board[3] and board[1] != ' '):
        return True
    elif (board[4] == board[5] and board[4] ==
board[6] and board[4] != ' '):
        return True
    elif (board[7] == board[8] and board[7] ==
board[9] and board[7] != ' '):
        return True
    elif (board[1] == board[4] and board[1] ==
board[7] and board[1] != ' '):
        return True
    elif (board[2] == board[5] and board[2] ==
board[8] and board[2] != ' '):
        return True
    elif (board[3] == board[6] and board[3] ==
board[9] and board[3] != ' '):
        return True
    elif (board[1] == board[5] and board[1] ==
board[9] and board[1] != ' '):
        return True
    elif (board[7] == board[5] and board[7] ==
board[3] and board[7] != ' '):
        return True
    else:
        return False

def checkWhichMarkWon(mark):
```

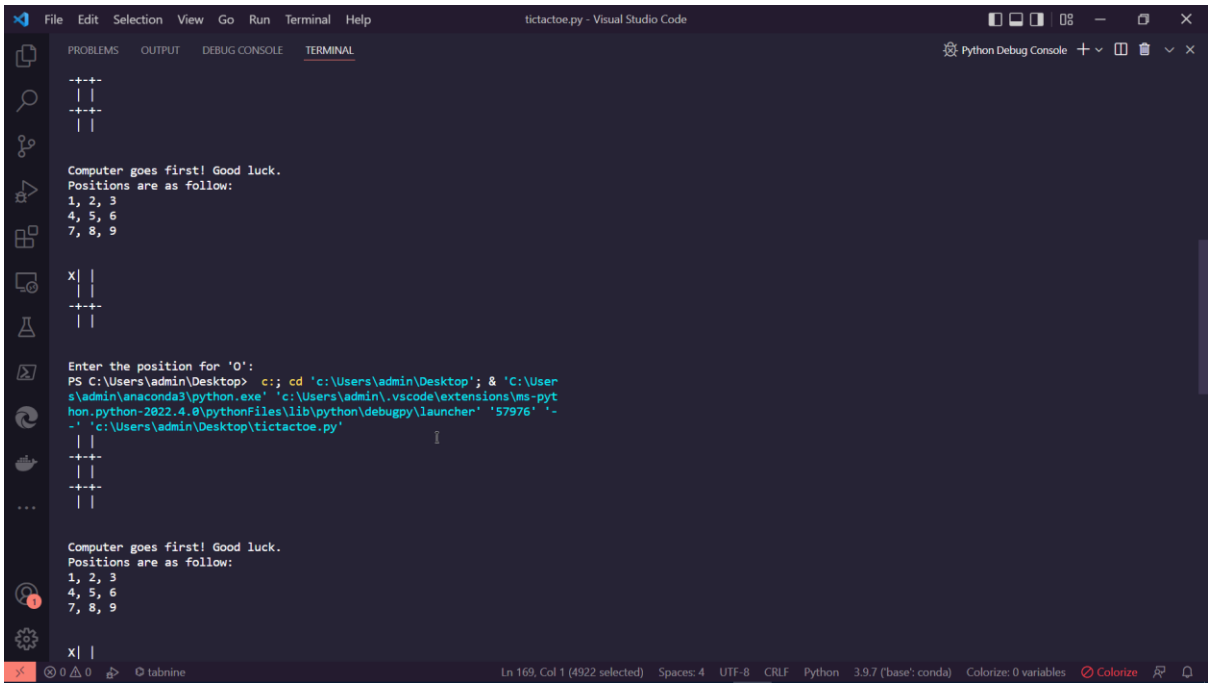
```
    if board[1] == board[2] and board[1] ==
board[3] and board[1] == mark:
        return True
    elif (board[4] == board[5] and board[4] ==
board[6] and board[4] == mark):
        return True
    elif (board[7] == board[8] and board[7] ==
board[9] and board[7] == mark):
        return True
    elif (board[1] == board[4] and board[1] ==
board[7] and board[1] == mark):
        return True
    elif (board[2] == board[5] and board[2] ==
board[8] and board[2] == mark):
        return True
    elif (board[3] == board[6] and board[3] ==
board[9] and board[3] == mark):
        return True
    elif (board[1] == board[5] and board[1] ==
board[9] and board[1] == mark):
        return True
    elif (board[7] == board[5] and board[7] ==
board[3] and board[7] == mark):
        return True
    else:
        return False

def playerMove():
    position = int(input("Enter the position for
'O':  "))
    insertLetter(player, position)
    return


def compMove():
    bestScore = -800
```

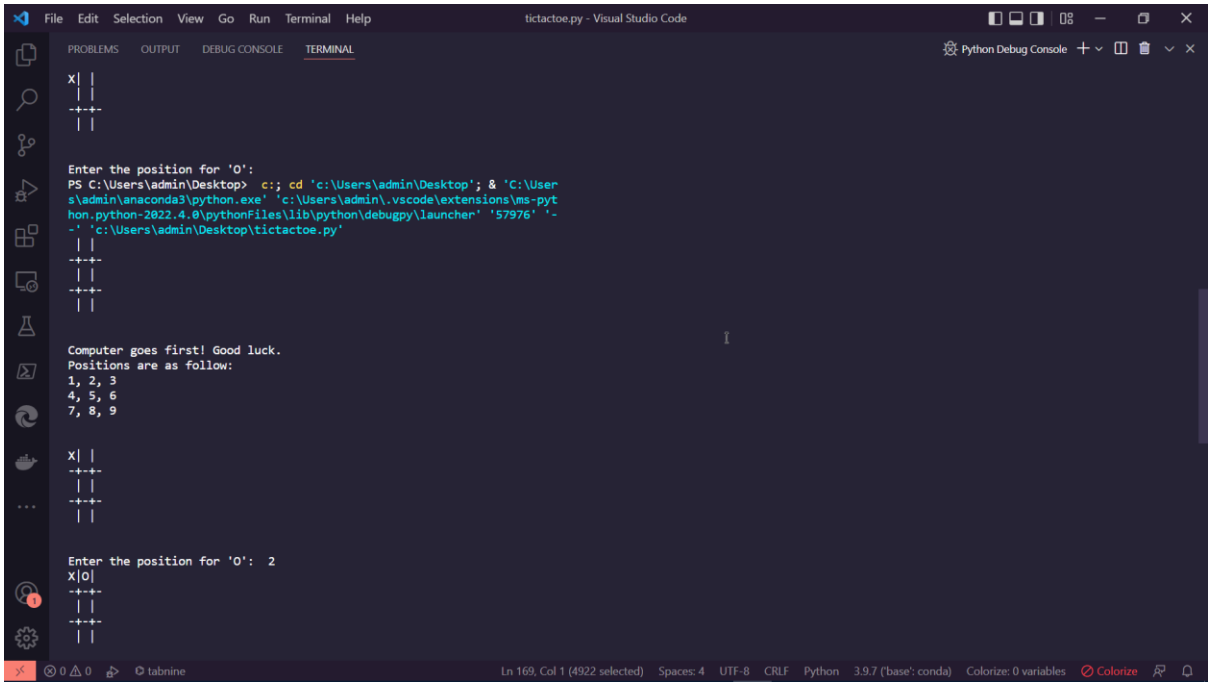
```
    bestMove = 0
    for key in board.keys():
        if (board[key] == ' '):
            board[key] = bot #bot will play if the
space is empty
            score = minimax(board, 0, False)
            board[key] = ' '
            if (score > bestScore):
                bestScore = score
                bestMove = key

    insertLetter(bot, bestMove)
    return

def minimax(board, depth, isMaximizing):
    if (checkWhichMarkWon(bot)):
        return 1
    elif (checkWhichMarkWon(player)):
        return -1
    elif (checkDraw()):
        return 0

    if (isMaximizing):
        bestScore = -800
        for key in board.keys():
            if (board[key] == ' '):
                board[key] = bot
                score = minimax(board, depth + 1,
False)
                board[key] = ' '
                if (score > bestScore):
                    bestScore = score
        return bestScore

    else:
```

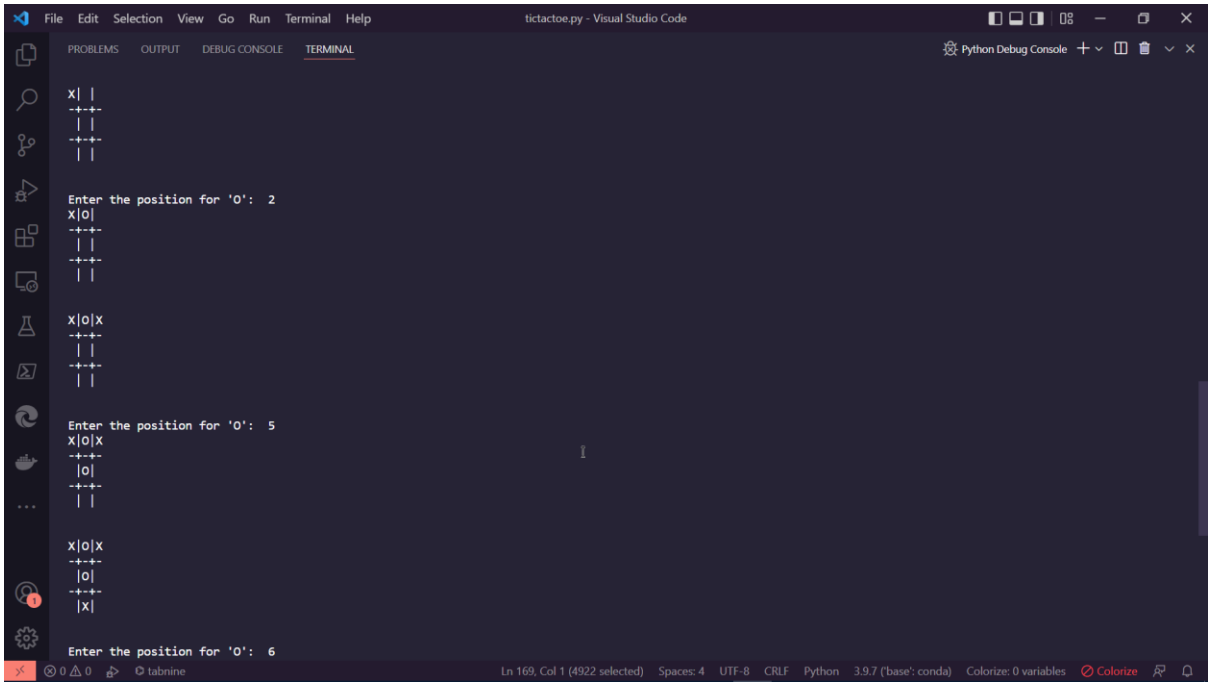
```
        bestScore = 800
        for key in board.keys():
            if (board[key] == ' '):
                board[key] = player
                score = minimax(board, depth + 1,
True)
                board[key] = ' '
                if (score < bestScore):
                    bestScore = score
        return bestScore


printBoard(board)
print("Computer goes first! Good luck.")
print("Positions are as follow:")
print("1, 2, 3 ")
print("4, 5, 6 ")
print("7, 8, 9 ")
print("\n")
player = 'O'
bot = 'X'


global firstComputerMove
firstComputerMove = True

while not checkForWin():
    compMove()
    playerMove()
```

Output:

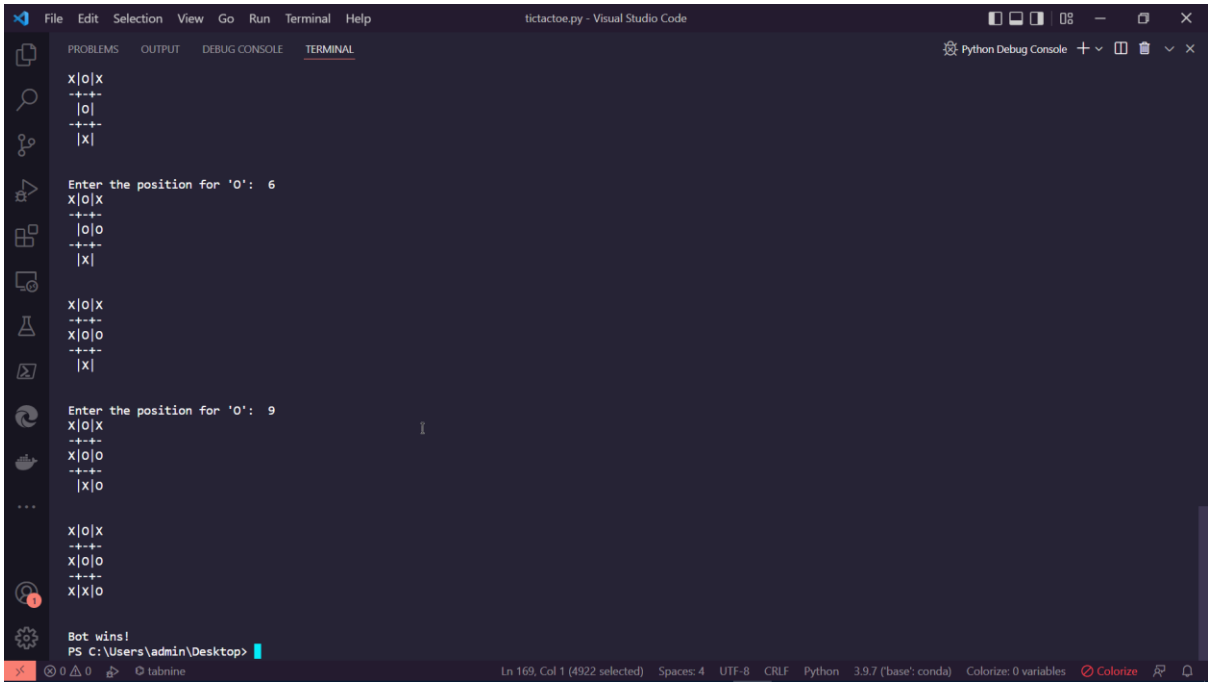

File Edit Selection View Go Run Terminal Help
tictactoe.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console
+ FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\admin\Desktop> & 'C:\Users\admin\anaconda3\python.exe' 'c:
0\pythonFiles\lib\python\debugpy\launcher' '57877' '--' 'c:\Users\admin
| |
-+-+-
| |
-+-+-
| |
Computer goes first! Good luck.
Positions are as follow:
1, 2, 3
4, 5, 6
7, 8, 9
X| |
| |
-+-+-
| |
Enter the position for 'O':
PS C:\Users\admin\Desktop> c:; cd 'c:\Users\admin\Des
ktop'; & 'C:\Users\admin\anaconda3\python.exe' 'c:\Use
rs\admin\.vscode\extensions\ms-python.python-2022.4.0\
pythonFiles\lib\python\debugpy\launcher' '57882' '--'
'c:\Users\admin\Desktop\tictactoe.py'
| |
-+-+-
| |
-+-+-
| |


File Edit Selection View Go Run Terminal Help
tictactoe.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console
-+-+-
| |
-+-+-
| |
Computer goes first! Good luck.
Positions are as follow:
1, 2, 3
4, 5, 6
7, 8, 9
X| |
| |
-+-+-
| |
Enter the position for 'O':
PS C:\Users\admin\Desktop> c:; cd 'c:\Users\admin\Desktop'; & 'C:\User
s\admin\anaconda3\python.exe' 'c:\Users\admin\.vscode\extensions\ms-pyt
hon.python-2022.4.0\pythonFiles\lib\python\debugpy\launcher' '57976' '-
-' 'c:\Users\admin\Desktop\tictactoe.py'
| |
-+-+-
| |
-+-+-
| |
Computer goes first! Good luck.
Positions are as follow:
1, 2, 3
4, 5, 6
7, 8, 9
X| |
tabnine
Ln 169, Col 1 (4922 selected) Spaces: 4 UTF-8 CRLF Python 3.9.7 ('base': conda) Colorize: 0 variables Colorize





Conclusion: Implementation of the game playing algorithm has been successful.


File Edit Selection View Go Run Terminal Help
tictactoe.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python Debug Console
x|o|x
-+-+-
|o|
-+-+-
|x|
Enter the position for 'O': 6
x|o|x
-+-+-
|o|o
-+-+-
|x|
x|o|x
-+-+-
x|o|o
-+-+-
|x|
Enter the position for 'O': 9
x|o|x
-+-+-
x|o|o
-+-+-
|x|o
x|o|x
-+-+-
x|o|o
-+-+-
x|x|o
Bot wins!
PS C:\Users\admin\Desktop>
tabnine
Ln 169, Col 1 (4922 selected) Spaces: 4 UTF-8 CRLF Python 3.9.7 ('base': conda) Colorize: 0 variables Colorize