Apurv Sarode
2019130054
Batch C
TE Comps

Experiment-3

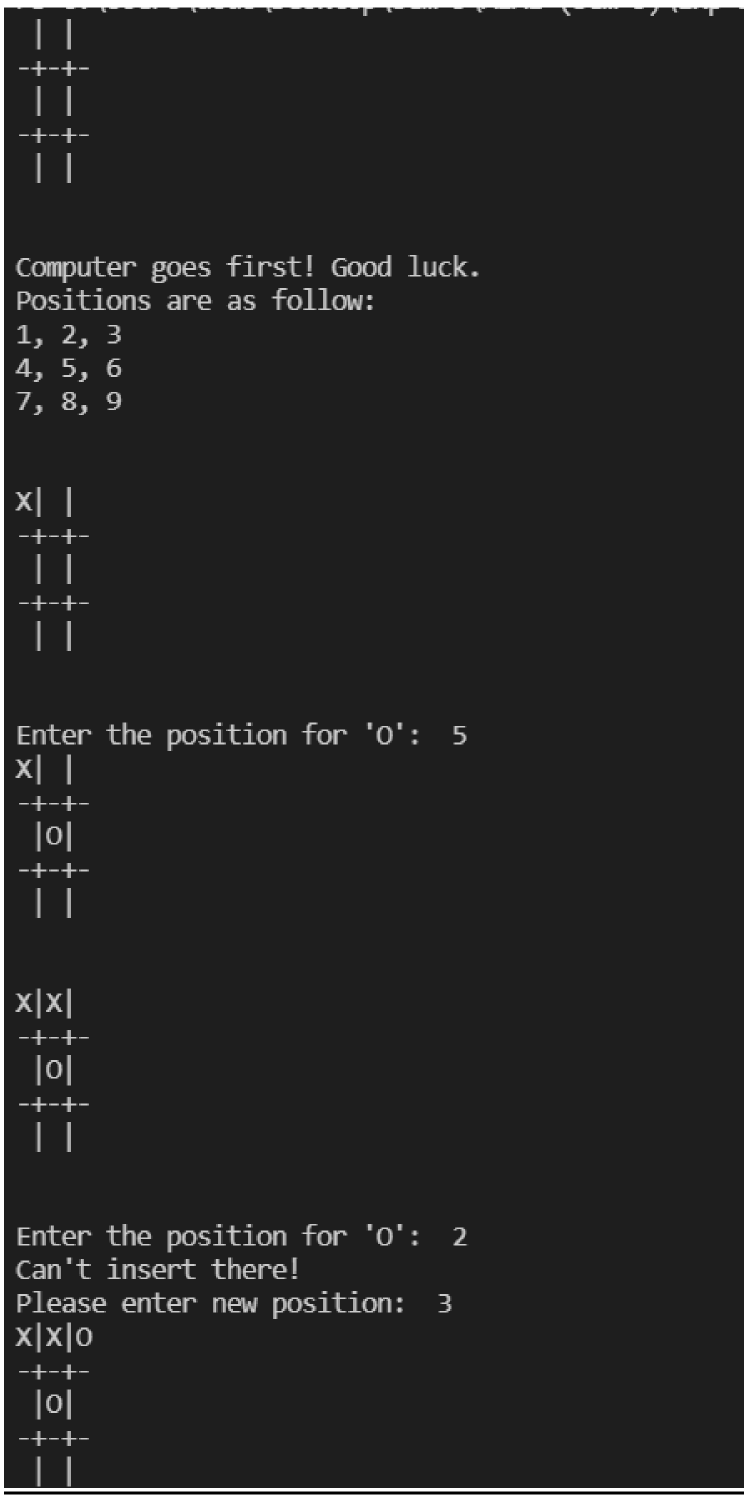
Aim: Implement TicTacToe using a* Algorithm

Code:

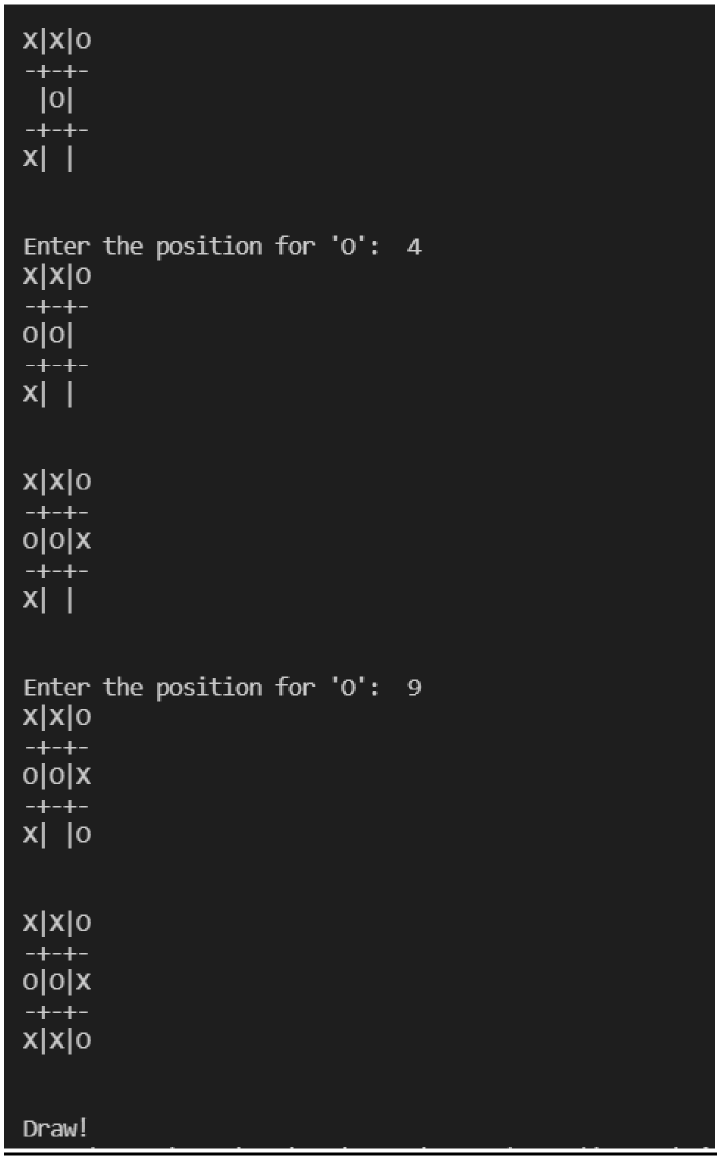
```python
def printBoard(board):
    print(board[1] + '|' + board[2] + '|' + board[3])
    print('-+-+-')
    print(board[4] + '|' + board[5] + '|' + board[6])
    print('-+-+-')
    print(board[7] + '|' + board[8] + '|' + board[9])
    print("\n")


def spaceIsFree(position):
    if board[position] == ' ':
        return True
    else:
        return False


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


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):
```

```python
        return True
    elif (board[7] == board[5] and board[7] == board[3] and board[7] == mark):
        return True
    else:
        return False


def checkDraw():
    for key in board.keys():
        if (board[key] == ' '):
            return False
    return True


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
            score = gamealgo(board, 0, False)
            board[key] = ' '
            if (score > bestScore):
                bestScore = score
                bestMove = key

    insertLetter(bot, bestMove)
    return


def gamealgo(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 = gamealgo(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 = gamealgo(board, depth + 1, True)
                board[key] = ' '
                if (score < bestScore):
                    bestScore = score
        return bestScore


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

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:



```
 | |
-+-+-
 | |
-+-+-
 | |


Computer goes first! Good luck.
Positions are as follow:
1, 2, 3
4, 5, 6
7, 8, 9


X| |
-+-+-
 | |
-+-+-
 | |


Enter the position for 'O':  5
X| |
-+-+-
 |O|
-+-+-
 | |


X|X|
-+-+-
 |O|
-+-+-
 | |


Enter the position for 'O':  2
Can't insert there!
Please enter new position:  3
X|X|O
-+-+-
 |O|
-+-+-
 | |
```



```
x|x|o
-+-+-
 |o|
-+-+-
x| |


Enter the position for 'O':  4
x|x|o
-+-+-
o|o|
-+-+-
x| |


x|x|o
-+-+-
o|o|x
-+-+-
x| |


Enter the position for 'O':  9
x|x|o
-+-+-
o|o|x
-+-+-
x| |o


x|x|o
-+-+-
o|o|x
-+-+-
x|x|o


Draw!
```

Conclusion:

The purpose of this experiment was to use the informed search approach to implement the tic-tac-toe game. To find the best place to put the 'X' or 'O,' I first calculated the difference between the winning combinations of bot(X) and user(O) for each choice in that round, then for the maximum values obtained, I calculated which choice would not lead to computer victory and which would lead to user victory by checking the number of moves required for victory for each choice and selecting the one with the fewest moves. If there are multiple movements that are comparable, the piece is placed at random. Download the tic-tac-toe.py file and run the python code to view the demonstration.