AI LAB 1-TIC TAC TOE

1BM21CS203

```
import random


# Initialize the game board

board = [' ' for _ in range(10)]



def insertLetter(letter, pos):

    global board

    board[pos] = letter



def spaceIsFree(pos):

    return board[pos] == ' '



def printBoard(board):

    # print(' | |')

    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])

    # print(' | |')

    print('-----------')

    # print(' | |')

    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])

    # print(' | |')

    print('-----------')

    # print(' | |')
```

```
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    # print(' | |')


def isWinner(bo, le):
    return (
        (bo[7] == le and bo[8] == le and bo[9] == le) or
        (bo[4] == le and bo[5] == le and bo[6] == le) or
        (bo[1] == le and bo[2] == le and bo[3] == le) or
        (bo[1] == le and bo[4] == le and bo[7] == le) or
        (bo[2] == le and bo[5] == le and bo[8] == le) or
        (bo[3] == le and bo[6] == le and bo[9] == le) or
        (bo[1] == le and bo[5] == le and bo[9] == le) or
        (bo[3] == le and bo[5] == le and bo[7] == le)
    )


def playerMove():
    global board
    run = True
    while run:
        move = input('Please select a position to place an \'X\' (1-9): ')
        try:
            move = int(move)
            if 1 <= move <= 9:
                if spaceIsFree(move):
                    run = False
```

```
                insertLetter('X', move)
            else:
                print('Sorry, this space is occupied!')
        else:
            print('Please type a number within the range!')
    except ValueError:
        print('Please type a number!')



def compMove():
    global board
    possibleMoves = [x for x, letter in enumerate(board) if letter == ' ' and x != 0]

    for let in ['O', 'X']:
        for i in possibleMoves:
            boardCopy = board[:]
            boardCopy[i] = let
            if isWinner(boardCopy, let):
                return i

    cornersOpen = [i for i in possibleMoves if i in [1, 3, 7, 9]]
    if cornersOpen:
        return selectRandom(cornersOpen)

    if 5 in possibleMoves:
        return 5
```

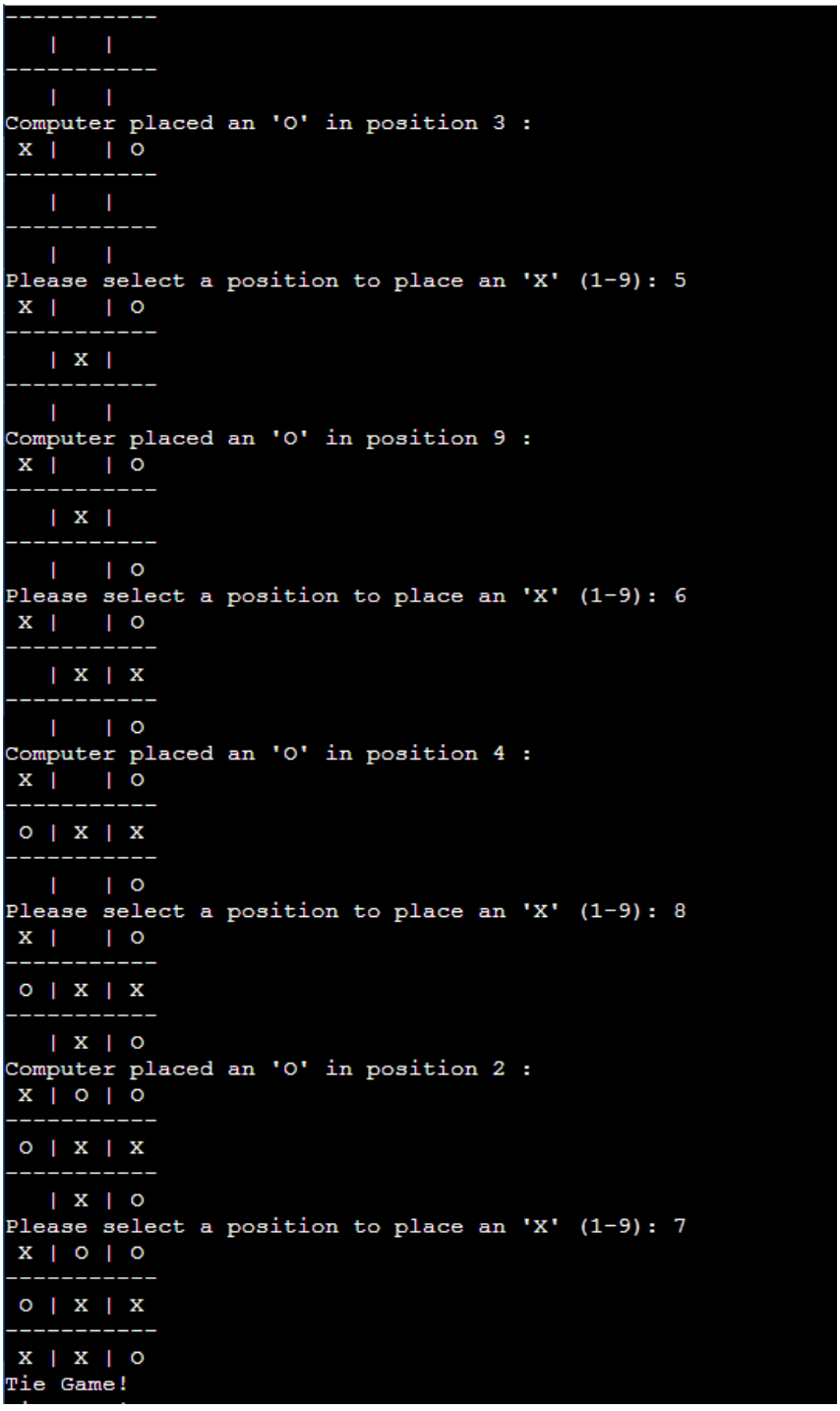
```
    edgesOpen = [i for i in possibleMoves if i in [2, 4, 6, 8]]
    if edgesOpen:
        return selectRandom(edgesOpen)

    return None  # Indicates a tie



def selectRandom(li):
    ln = len(li)
    r = random.randrange(ln)
    return li[r]



def isBoardFull(board):
    return board.count(' ') <= 1



def main():
    global board
    print('Welcome to Tic Tac Toe!')
    printBoard(board)

    while not isBoardFull(board):
        if not isWinner(board, 'O'):
            playerMove()
            printBoard(board)
        else:
```

```
            print('Sorry, O\'s won this time!')
            break

        if not isWinner(board, 'X'):
            move = compMove()
            if move is None:
                print('Tie Game!')
            else:
                insertLetter('O', move)
                print('Computer placed an \'O\' in position', move, ':')
                printBoard(board)
        else:
            print('X\'s won this time! Good Job!')
            break

    if isBoardFull(board):
        print('Tie Game!')

    while True:
        answer = input('Do you want to play again? (Y/N)')
        if answer.lower() == 'y' or answer.lower() == 'yes':
            board = [' ' for _ in range(10)]
            print('-----------------------------------')
            main()
        else:
            break
```

Run the game

main()

OUTPUT:



ANALYSIS:

The code checks for winning conditions through the isWinner function. This function takes the current state of the board and a letter ('X' or 'O') and checks if that letter has won the game.

The winning conditions are checked for each row, column, and diagonal of the board. If any of these conditions are met, the function returns True, indicating that the specified letter has won. The winning combinations are:

Horizontal rows: [7, 8, 9], [4, 5, 6], [1, 2, 3]

Vertical columns: [7, 4, 1], [8, 5, 2], [9, 6, 3]

Diagonals: [7, 5, 3], [9, 5, 1]

If none of these winning conditions are met, the function returns False.