1. Implement Tic -Tac -Toe Game.

```python
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')

def spaceFree(pos):
    if(board[pos]==' '):
        return True
    else:
        return False

def checkWin():
    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[5] and board[1] == board[9] and board[1] != ' '):
        return True
    elif (board[3] == board[5] and board[3] == board[7] and board[3] != ' '):
        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
    else:
        return False

def checkMoveForWin(move):
    if (board[1]==board[2] and board[1]==board[3] and board[1] ==move):
        return True
```

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


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



def insertLetter(letter, position):
    if (spaceFree(position)):
        board[position] = letter
        printBoard(board)

        if (checkDraw()):
            print('Draw!')
        elif (checkWin()):
            if (letter == 'X'):
                print('Bot wins!')
            else:
                print('You win!')
        return

    else:
        print('Position taken, please pick a different position.')
        position = int(input('Enter new position: '))
        insertLetter(letter, position)
        return

player = 'O'
bot ='X'
```

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

def compMove():
    bestScore=-1000
    bestMove=0
    for key in board.keys():
        if (board[key]==' '):
            board[key]=bot
            score = minimax(board, False)
            board[key] = ' '
            if (score > bestScore):
                bestScore = score
                bestMove = key

    insertLetter(bot, bestMove)
    return


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

    if isMaximizing:
        bestScore = -1000

        for key in board.keys():
            if board[key] == ' ':
                board[key] = bot
                score = minimax(board, False)
                board[key] = ' '
                if (score > bestScore):
                    bestScore = score
        return bestScore
    else:
        bestScore = 1000

        for key in board.keys():
            if board[key] == ' ':
                board[key] = player
                score = minimax(board, True)
```

```
                board[key] = ' '
                if (score < bestScore):
                    bestScore = score
        return bestScore

while not checkWin():
    playerMove()
    compMove()
print("Varsha P(1BM22Cs320)")
```

Output

```
Enter position for O:1
O| |
-+-+-
 | |
-+-+-
 | |


O| |
-+-+-
 |x|
-+-+-
 | |


Enter position for O:2
O|O|
-+-+-
 |x|
-+-+-
 | |


O|O|x
-+-+-
 |x|
-+-+-
 | |


Enter position for O:4
O|O|x
-+-+-
O|x|
-+-+-
 | |


O|O|x
-+-+-
O|x|
-+-+-
x| |


Bot wins!
Varsha P(1BM22CS320)
```

Observation

① | Tic Tac Toe Game |

Function minimax (node, depth, is Maximizing Play)
if node is a terminal state:
return evaluate (node)

if isMaximizing Player:
        best Value = -infinity
        for Each child in node:
        value = minimax (child, dypth+1, false)
        best Value = max (best Value, value)
        return best Value

else:
        best Value = + infinity
        for each child in node:
        value = minimax (child, dypth+1, true)
        best Value = min (best Value, value)
        return Best Value.