AI Lab Test

Nilesh Gangopadhyay
1BH18CS088
5th - B

Q Implement Tic - Tac - Toe using 2-agent algorithm (Computer Vs computer)

So, hence we can see that the automatic implementation of Tic-Tac-toe can solve our problem

So, hence we can do it with using the random only

Code

```
import numpy as np
import random
from time import sleep

# To creates the empty board
   def create_board():
         return (np.array ([0,0,0],
                           [0,0,0],
                           [0,0,0]]))

def possibilities (board):
      l = []
      for i in range (len (board)):
           for j in range (len (board)):
                    if board[i][j] == 0
                       l.append((i,j))

   return (l)
```

Nilesh

Neelesh Gangopadhyay
1BM18CS058

```
# to select the random place only.
def random-place (board, player):
        Selection = possibilities (board)
        current_loc = random. choice (selection)
         board [current_loc]= player
        return (board)

# to check if the player has their mark on the horizontal row
def row_win (board, player):
        for x in range (len(board)):
            win = True

            for y in range (len(board)):
                if board [x,y] ! = player:
                win = False
              continue
        if win == True:
            return (win)

    return (win)

def col-win (board, player):
        for x in range (len(board)):
            win = True
        for y in range[ len (board)):
            if board [y)[x) ! = player.
               win = false
            continue
```

Neele

Nilesh Gangopadhyay
1BM18CS058

```
if win == True:
    return (win)
    return (win)

# To check whether the diagonal win or not
    def diag-win (board, player):
        win = True
        y = 0
        for x in range (len(board)):
            if board [x,x] != player :
                win = False
        if win:
            return win
        win = True
        if win:
            for x in range (len(board)):
                y = len(board)-1-x
                if board [x,y) != player:
                    win = False

        return win
# to evaluate whether there is a winner or a tie
    def evaluate (board):
        winner = 0

        for player in [1,2]:
            if (row-win(board, player)
print ("board after " + str (count) + " move")
```

Neel

(3)

```
    print (board)
    Sleep (2)
     counter += 1
      winner = evaluate (board)
   if winner != 0:
         break
  return (winner)

print ("winner is: " + str (play_game()))
```