PAGE NO :
DATE : 24/9/24

TicTacToe

Algorithm

S1) ~~Assign "X" or "O" at random for the 1st move~~

S2) ~~Take user input for array location, and change turn.~~

S3) ~~An empty 3x3~~

S1) create a 3x3 empty board and ~~win cond^n~~
board = [["-"]*3 for i in range(3)]
~~win_cond = {(0~~

S2) Assign "X" or "O" to player and player moves first.

S2.5) User makes move.

S3) If centre is empty system places symbol at the centre.

S3.5) Check for system win cond^n if present place and stop.

S4) For system move check for wins conditions if any cond^n ~~met~~ make a move to stop, else random, multiple win cond^n random

S5) Check is board is full, if yes stop game play. else → S2.5)

PAGE NO :
DATE :

Code :-

```
import random
import numpy as np

board = [[" - "] * 3 for i in range(3)]

def check_win():
    for i in range(3):
        if (board[i][0] == board[i][1] == board[i][2]) != " ":
            return True
        if (board[0][i] == board[1][i] == board[2][i]) != " ":
            return True

    if board[0][0] == board[1][1] == board[2][2] != " ":
        return True
    if board[0][2] == board[1][1] == board[2][0] != " ":
        return True

    return False.

def full():
    return all(cell != " - " for row in board for cell in row)

def can_win(m):
    for i in range(3):
        row = board[i]
        if row.count(m) == 2 and row.count(" - ") == 1:
            return (i, row.index(" - "))

    for i in range(3):
        col = [board[j][i] for j in range(3)]
        if col.count(m) == 2 and col.count(" - ") == 1:
            return (col.index(" - "), i)
```

```
d1 = [board[i][i] for i in range(3)]
if d1.count(m) == 2 and d1.count(" ") == 1:
    return (d1.count(" "), d1.count(" "))

d2 = [board[i][2-i] for i in range(3)]
if d2.count(m) == 2 and d2.count(" ") == 1:
    return (d2.count(" "), 2 - d2.count(" "))

return None

def display():
    print(np.array(board))

while True:
    display()
    u = tuple(map(int, input("Enter row and col for X").strip().split()))

    if board[u[0]][u[1]] != " ":
        print("Occupied already")
        continue

    board[u[0]][u[1]] = "X"

    if checkwin():
        display()
        print("X wins!")
        break

    if full():
        display()
        print("Its a tie!")
        break
```

```
move = can_win("O")
if move is None:
    move = can_win("X")
    if move is None:
        empty = [(i, j) for i in range(3)
                        for j in range(3)
                        if board[i][j] == "-"]
        move = random.choice(empty)
if board[1][1] == "-":
    move = (1, 1)
board[move[0]][move[1]] = "O"

if check_win():
    display()
    print("O wins")
    break
```