Date 24/09/2024
Page 1

24/09

Tic - Tac - Toe

Algorithm:

Step 1: Initilize a variable flag to run a loop continuously. flag = True

Step 2: Create a 3 x 3 grid and initialize the symbol "-" as empty spaces
d = [['-','-','-'], ['-','-','-'], ['-','-','-']]

Step 3: Let the user start the game and mark 'X' and the computer mark 'O' at random empty spaces, player (preferrably at middle)

Step 4: Checks all the win conditions and marks accordingly i.e,

Step 5: Check if the symbol 'X' is present in the same row or same column or diagonal.
If true, mark 'O' in the respective row, column, diagonal.
else
mark 'O' at random.

Step 6: If no empty space
print ("tie")

Step 7: If three "O" are present in same row (or) same column (or) diagonal,
print ("computer wins")

Date __/__/__
Page _____

Program:

```
import numpy as np
import random

flag = True
d = [["-"]*3 for i in range(3)]
win = ""

def checkr(i):
    if d[i][0] == d[i][1] == d[i][2] != '-':
        global win
        win = d[i][0]
        return True
    return False

def checkc(i):
    if d[0][i] == d[1][i] == d[2][i] != '-':
        global win
        win = d[0][i]
        return True
    return False.

def checkd(i):
    global win
    if d[0][0] == d[1][1] == d[2][2] != '-':
        win = d[0][0]
        return True
    if d[0][2] == d[1][1] == d[2][0] != '-':
        win = d[0][2]
        return True
    return False
```

Date __/__/__
Page _____

```
def display():
    for row in d:
        print(" ".join(row))

def check():
    l = set(d[0] + d[1] + d[2])
    if "-" not in l:
        return True

def computer_move():
    for i in range(3):
        if d[i].count("o") == 2 and d[i].
        count("-") == 1:
            return (i, d[i].index("-"))
        if d[i].count("x") == 2 and d[i].
        count("-") == 1:
            return (i, d[i].index("-"))

    for i in range(3):
        col = [d[0][i], d[1][i], d[2][i]]
        if col.count("o") == 2 and col.
        count("-") == 1:
            return (col.index("-"), i)
        if col.count("x") == 2 and col.
        count("-") == 1:
            return (col.index("-"), i)

diag1 = [d[0][0], d[1][1], d[2][2]]
diag2 = [d[0][2], d[1][1], d[2][0]]

if diag1.count("o") == 2 and diag1.
count("-") == 1:
    return (diag1.index("-"), diag1.index
("-"))
```

```
if diag1.count("X") == 2 and diag1.
    count("-") == 1:
        return (diag1.index("-"), diag1.
        index("-"))

if d[0][1] == "-":
    return (1, 0)

for (i, j) in [(0,0), (0,2), (2,0), (2,2)]:
    if d[i][j] == "-":
        return (i, j)

for i in range(3):
    for j in range(3):
        if d[i][j] == "-":
            return (i, j)

turn = random.choice(["X", "O"])
print("{turn} goes first!")

while flag:
    if turn == "X":
    x = tuple(map(int, input("Enter
    row and column of X input: ",
    strip().split()))
    if d[x[0]][x[1]] != '-':
        print("Spot already occupied,
        pick a new location.")
        continue.

    d[x[0]][x[1]] = "X"

display()
```

Date __/__/__
Page _____

```
if checkd() or any (checkr(i) for i in
range(3)) or any (checkc(i) for i in
range(3)):
    print(f"{turn} Wins!")
break()

if check():
    print("Tie")
    break

turn = "O"

elif turn == "O":
print("Computer is thinking...")
y = computer move()
d[y[0]][y[0]] = "O"

if checkd() or any (checkr(i) for i
    in range(3)) or any (checkc(i) for
    i in range(3)):
    print(f"{turn} Wins!")
    display
    break

display
if check:
    print("Tie")
    break
turn = "X"
```