

Name: - L Prathyusha

In [1]:

```
# importing all necessary libraries
import numpy as np
import random
from time import sleep
```

In [2]:

```
# Creates an empty board
def create_board():
    return(np.array([[0, 0, 0],
                    [0, 0, 0],
                    [0, 0, 0]]))
```

In [3]:

```
# Check for empty places on board
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)
```

In [4]:

```
# Select a random place for the player
def random_place(board, player):
    selection = possibilities(board)
    current_loc = random.choice(selection)
    board[current_loc] = player
    return(board)
```

In [5]:

```
# Checks whether the player has three
# of their marks in a 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)
```

In [6]:

```
# Checks whether the player has three  
# of their marks in a vertical row  
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  
  
        if win == True:  
            return(win)  
    return(win)
```

In [7]:

```
# Checks whether the player has three  
# of their marks in a diagonal row  
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
```

In [8]:

```
# Evaluates whether there is  
# a winner or a tie  
def evaluate(board):  
    winner = 0  
  
    for player in [1, 2]:  
        if (row_win(board, player) or  
            col_win(board, player) or  
            diag_win(board, player)):  
  
            winner = player  
  
    if np.all(board != 0) and winner == 0:  
        winner = -1  
    return winner
```

In [9]:

```
# Main function to start the game
def play_game():
    board, winner, counter = create_board(), 0, 1
    print(board)
    sleep(2)

    while winner == 0:
        for player in [1, 2]:
            board = random_place(board, player)
            print("Board after " + str(counter) + " move")
            print(board)
            sleep(2)
            counter += 1
            winner = evaluate(board)
            if winner != 0:
                break
    return(winner)
```

In [10]:

```
# Driver Code
print("Winner is: " + str(play_game()))
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 1 0]
 [0 0 0]
 [0 0 0]]
Board after 2 move
[[0 1 0]
 [0 0 2]
 [0 0 0]]
Board after 3 move
[[0 1 0]
 [0 1 2]
 [0 0 0]]
Board after 4 move
[[2 1 0]
 [0 1 2]
 [0 0 0]]
Board after 5 move
[[2 1 0]
 [0 1 2]
 [0 1 0]]
Winner is: 1
```

In []:

