Ex.No. 1 **Automatic Tic Tac Toe Game Using Random Number**
Date:

Aim:

To write a python program to implement automatic tic-tac-toe game using random number.

Algorithm:

Step 1: Start

Step 2: Create a 3x3 board and initialize it with 0

Step 3: For each player i.e. 1 or 2, choose a position on the board randomly and mark the location with the player's number

Step 4: Print the board after each move

Step 5: Evaluate the board after each move to check whether a row or column or diagonal has the same player number. If so display the winner's name

Step 6: Repeat steps 2 through 5 until aa winner emerges or all the nine positions are marked

Step 7: If there is no winner after all the nine moves, then display -1

Step 8: Stop

Program:

```
import numpy as np
import random
from time import sleep
 # Creates an empty board
  def create_board():
   return(np.array([[0, 0, 0],
                    [0, 0, 0],
                    [0, 0, 0]]))
 # 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)
# 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)
# 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)
# 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)
```

```
# 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

# 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

# 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)

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

Output:

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 0 0]
 [0 0 0]
 [1 0 0]]
Board after 2 move
[[0 0 0]
 [0 2 0]
 [1 0 0]]
```

Board after 3 move

[[0 1 0]

[0 2 0]

[1 0 0]]

Board after 4 move

[[0 1 0]

[2 2 0]

[1 0 0]]

Board after 5 move

[[1 1 0]

[2 2 0]

[1 0 0]]

Board after 6 move

[[1 1 0]

[2 2 0]

[1 2 0]]

Board after 7 move

[[1 1 0]

[2 2 0]

[1 2 1]]

Board after 8 move

[[1 1 0]

[2 2 2]

[1 2 1]]

Winner is: 2

Result:

Thus, the Python program to implement automatic tic-tac-toe game using random number was executed and the output was verified successfully.

Ex.No. 2 **Drug Screening**

Date:

Aim:

To write a python program to implement drug screening.

Algorithm:

Step 1: Start.

Step 2: Define the function 'drug_user.' The function takes the following parameters: probability threshold, sensitivity, specificity, prevalence, and verbose (default set to `True`).

Step 3: Calculate the probability of being a drug user (`p_user`) as the given `prevalence`.

Step 4: Calculate the probability of not being a drug user (`p_non_user`) as `1 - prevalence`.

Step 5: Calculate the probability of testing positive given the person is a drug user (`p_pos_user`) as the given `sensitivity`.

Step 6: Calculate the probability of testing negative given the person is a drug user (`p_neg_user`) as the given `specificity`.

Step 7: Calculate the probability of testing positive given the person is not a drug user (`p_pos_non_user`) as `1 - specificity`.

Step 8: Calculate the numerator (`num`) as `p_pos_user * p_user`.

Step 9: Calculate the denominator (`den`) as `p_pos_user * p_user + p_pos_non_user * p_non_user`.

Step 10: Calculate the probability (`prob`) as `num / den`.

Step 11: If `verbose` is `True`, print "The test-taker could be a user" if `prob` is greater than `prob_th`, otherwise print "The test-taker may not be a user".

Step 12: Return the probability `prob`. Call the `drug_user` function with the given parameters.

Step 13: Assign the returned probability to variable `p`.

Step 14: Print "Probability of the test taker being a drug user is" followed by the rounded value of `p` (rounded to 3 decimal places).

Step 15: Stop

Program:

```
def drug_user(
        prob_th=0.5,
        sensitivity=0.99,
        specificity=0.99,
        prevelance=0.01,
        verbose=True):
    """
    Computes the posterior using Bayes' rule
    """
    p_user = prevelance
    p_non_user = 1-prevelance
    p_pos_user = sensitivity
    p_neg_user = specificity
    p_pos_non_user = 1-specificity

    num = p_pos_user*p_user
    den = p_pos_user*p_user+p_pos_non_user*p_non_user

    prob = num/den

    if verbose:
        if prob > prob_th:
            print("The test-taker could be an user")
        else:
            print("The test-taker may not be an user")

    return prob

p = drug_user(prob_th=0.5, sensitivity=0.97, prevelance=0.005)
print("Probability of the test taker being a drug user is", round(p,3))
```

Output:

The test-taker may not be an user
Probability of the test taker being a drug user is 0.328

Result:

Thus, the Python program to implement drug screening was executed and the output was verified successfully.