

SRM INSTITUTE OF SCIENCE & TECHNOLOGY DEPARTMENT OF NETWORKING & COMMUNICATIONS **18CSC305J-ARTIFICIAL INTELLIGENCE**

SEMESTER – 6 BATCH-2

|  |  |
| --- | --- |
| **REGISTRATION NUMBER** | **RA1911028010018** |
| **NAME** | VAIBHAV SHARMA |

# B.Tech- CSE / CC, Third Year (Section: H2)

**Faculty In charge: Dr. S. Prabakeran, B.Tech, M.E, PhD**

**Assistant Professor**

**School of Computing - Department of**

**Networking and Communications**

**Year 2021-2022 / Even Semester**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ex No** | **DATE** | **Title** | **Page No** | **Marks** |
| 1 | 13/1/2022 | Toy Problem: Tic Tac Toe |  |  |

## Exercise: 2

**Date : 06/01/2022**

**Tic Tac Toe**

**Problem Statement :** The game Tic Tac Toe is also known as Noughts and Crosses or Xs and Os, the player needs to take turns marking the spaces in a 3x3 grid with their own marks, if 3 consecutive marks (Horizontal, Vertical,Diagonal) are formed then the player who owns these moves get won.

## Algorithm :

1. Create a 3x3 matrix
2. Check the diagonal conditions
3. Check the row conditions
4. Check the column conditions
5. The player who matches these conditions wins the game.

**Optimization technique :**

The idea is not to check every element in right and left diagonal instead use property of diagonals:

1.The sum of i and j is constant and unique for each right diagonal where i is the row of elements and j is the column of elements.

2.The difference of i and j is constant and unique for each left diagonal where i and j are row and column of element respectively.

**Tool :** Google Colab and Python 3.9.0

## Programming code :

# Tic-Tac-Toe Program using

# random number in Python

# importing all necessary libraries

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:

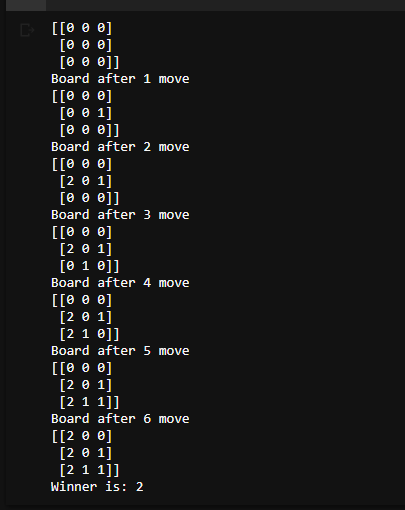
        break

  return(winner)

# Driver Code

print("Winner is: " + str(play\_game()))

## Output screen shots :



**Result : Tic tac toe game was conducted and the winning player was identified.**