**Name: Gaurang Vaghela**

**Rollno: TEAD-22561**

**ANN (SL-II)**

**Practical 9  
Problem Statement:** Design a Hopfield Network which stores 4 vectors

**Code:**

import numpy as np

class HopfieldNetwork:

def \_\_init\_\_(self, size):

self.size = size

self.weights = np.zeros((size, size))

def train(self, patterns):

for p in patterns:

p = p.reshape(self.size, 1)

self.weights += np.dot(p, p.T)

np.fill\_diagonal(self.weights, 0) # No self-connection

def recall(self, pattern, steps=5):

pattern = pattern.copy()

for \_ in range(steps):

for i in range(self.size):

raw = np.dot(self.weights[i], pattern)

pattern[i] = 1 if raw >= 0 else -1

return pattern

# Define 4 binary patterns (use -1 and 1 instead of 0 and 1)

patterns = np.array([

[1, -1, 1, -1, 1, -1],

[-1, 1, -1, 1, -1, 1],

[1, 1, -1, -1, 1, 1],

[-1, -1, 1, 1, -1, -1]

])

# Create and train Hopfield Network

hopfield\_net = HopfieldNetwork(size=6)

hopfield\_net.train(patterns)

# Test recall from a noisy version of the first pattern

test\_pattern = np.array([1, -1, 1, -1, -1, -1]) # Slightly different

recalled = hopfield\_net.recall(test\_pattern)

# Output

print("Original Pattern (noisy):", test\_pattern)

print("Recalled Pattern: ", recalled)

**Output:**

****