Ganpat University
॥ विद्यया समाजोत्कर्षः ॥
Institute of Computer Technology

Name: Tushar Panchal

En.No: 21162101014

Sub: FP(Functional Programing)

Branch: CBA

Batch:41

----------PRACTICAL 05----------

❖ ***Question-1 :***

(Simulation: coupon collector's problem) Coupon Collector is a classic statistics problem with many practical applications. The problem is to pick objects from a set of objects repeatedly and find out how many picks are needed for all the objects to be picked at least once. A variation of the problem is to pick cards from a shuffled deck of 52 cards repeatedly and find out how many picks are needed before you see one of each suit. Assume a picked card is placed back in the deck before picking another. Write a program to simulate the number of picks needed to get four cards, one from each suit and display the four cards picked (it is possible a card may be picked twice). Here is a sample run of the program:

Queen of Spades
5 of Clubs
Queen of Hearts
4 of Diamonds
Number of picks: 12

✓ Source Code :

```
from random import randint
def pick_card():
    values = ['Ace', 2, 3, 4, 5, 6, 7, 8, 9, 10, 'Jack', 'Queen', 'King']
    suits = ['Hearts', 'Diamonds', 'Spades', 'Clubs']
    card = randint(0, 51)
    card_val = card % 13
    card_suit = int(card/13)
    return [suits[card_suit], values[card_val]]

picks = 0
suits_picked = []
number_picked = []
```

```
while len(suits_picked) != 4:
    card = pick_card()
    if card[0] not in suits_picked:
        suits_picked.append(card[0])
        number_picked.append(card[1])
        picks += 1
print(f'Number of picks: {picks}')
for i in range(0, 4):
    print(number_picked[i], ' of ', suits_picked[i])
print('\n')
```

✓ **Output :**

```
Number of picks: 4
8  of  Spades
9  of  Diamonds
10  of  Hearts
Jack  of  Clubs
```

❖ ***Question-2 :***

A scientist is implementing an algorithm that passes through the list. On each pass, successive neighboring pairs are compared. If a pair is in decreasing order, its values are swapped; otherwise, the values remain unchanged. Gradually values bubble their way to the top and the larger values sink to the bottom. Help the scientist to test program that reads in ten numbers, invokes the function, and displays the resultant numbers.

✓ **Source Code :**

```
size = int(input('Enter number of elements: '))
num = [int(x)
       for x in input(f'Enter {size} space  separated numbers: ').split()]
print('Entered List: ', num)
for i in range(size):
    for j in range(0, size-i-1):
        if num[j] > num[j+1]:
            num[j], num[j+1] = num[j+1], num[j]
print('Sorted List: ', num)
```

✓ **Output :**

```
Enter number of elements: 4
Enter 4 space  separated numbers: 22 15 50 35
Entered List:  [22, 15, 50, 35]
Sorted List:  [15, 22, 35, 50]
```

❖ ***Question-3 :***

Given a matrix with N rows and M columns, the task is to check if the matrix is a Binary Matrix. A binary matrix is a matrix in which all the elements are either 0 or 1.

Input Format: The first line of the input contains two integer number N and M which represents the number of rows and the number of columns respectively, separated by a space. From the second line, take N lines input with each line containing M integer elements with each element separated by a space (not compulsory).

Output Format: Print ‘YES’ or ‘NO’ accordingly.

✓ Source Code :

```
def is_ident_mat(rows, cols, mat):
    if rows != cols:
        return False
    for i in range(rows):
        for j in range(cols):
            if i == j and mat[i][j] != 1:
                return False
            elif i != j and mat[i][j] != 0:
                return False
            return True


def is_bin_mat(rows, cols, mat):
    for i in range(rows):
        for j in range(cols):
            if mat[i][j] not in [0, 1]:
                return False
            return True
rows = int(input("Enter rows: "))
cols = int(input("Enter columns: "))
mat = [[] for i in range(rows)]
for i in range(rows):
    inp = input(f"Enter the elements of row {i + 1}: ").split()
    for j in range(cols):
        mat[i].append(int(inp[j]))
for i in range(rows):
    print(mat[i])

ident_mat = 'YES' if is_ident_mat(rows, cols, mat) else 'NO'
bin_mat = 'YES' if is_bin_mat(rows, cols, mat) else 'NO'
print('Indentity Matrix: ', ident_mat)
print('Binary Matrix: ', bin_mat)
```

✓ Output :

```
Enter rows: 2
Enter columns: 3
Enter the elements of row 1: 1 1 1
Enter the elements of row 2: 1 0 1
[1, 1, 1]
[1, 0, 1]
Indentity Matrix:  NO
Binary Matrix:  YES
```

```
Enter rows: 3
Enter columns: 3
Enter the elements of row 1: 1 0 0
Enter the elements of row 2: 0 1 0
Enter the elements of row 3: 0 0 1
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]
Indentity Matrix:  YES
Binary Matrix:  YES
```