TASK 4. 20/8/25

4.1 : Given a matrix mat where every row is sorted in strictly increasing order. return the smallest Common element in all rows.

AIM: To write a python programming that takes a sorted matrix (Each row sorted in strictly increasing order) as input from the user and finds the smallest element that is Present in all rows of matrix, if such element exists,

Algorithm

1. start
2. input the no.of rows (rows) and Columns (cols) from the user.
3. initialize an Empty matrix mat.
4. For Each row from 0 to rows -1.
5. Define a function Smallest Common Element (mat) to:
   * Let rows = total no.of rows, cols = total no.of Columns.
   * For each other row (starting from the 2nd row):
     1. Assume found = True.
     2. For Each other row
        - Perform binary search for num in row:
          - set left=0, right = cols -1.
          - while left <= right:
            - Find mid = (left+ right) //2.
            - else, move to the left half (right = mid -1).
3. if found is Still True after checking all rows, return num (Smallest Common element).
   - if no elements is found in all rows return -1.

Program:

```
def smallestCommonElement(mat):
    rows = len(mat)
    cols = len(mat[0])
    #check each number in first row
    for num in mat[0]:
        found = True
        for r in range(1, rows):
            #Binary search in Each row
            left, right = 0, cols - 1
            while left <= right:
                mid = (left + right) // 2
                if mat[r][mid] == num:
                    break
                elif mat[r][mid] < num:
                    left = mid + 1
                else:
                    right = mid - 1
            else: # executed if while loop doesn't break
                found = False
                break
        if found:
            return num
    return -1

#Get matrix input from user
rows = int(input("Enter no. of rows: "))
cols = int(input("Enter no. of columns: "))
```

```
mat = []
Print ("Enter matrix values row by row (sorted in increasing order):")
for _ in range(rows):
    row = list(map(int, input().split()))
    mat.append(row)
# Find and display smallest common element
result = SmallestCommonElement(mat)
Print ("smallest Common element in all rows:", result)
```

6. Call the function with mat as input.

7. Print the result

8. stop

Example 1:

Input:

```
4 5
1 2 3 4 5
2 4 5 8 10
3 5 7 9 11
1 3 5 7 9
```

output:

```
5
```

RESOLT: Thus, the smallest element that present in all rows of matrix is Successfully executed in the Element Exist.

4.2 Given an integer n, return an list of length n+1 such that for each i (0<=i <=n), ans[i] is the number of 1's in the binary representation of i.

AIM: To read a non-negative integer n from the user and generate a list of length n+1 where each element contains the count of 1s in the binary representation of its index.

Algorithm:

1. Start the program
2. Read the integer n from the user.
3. initialize an Empty list ans.
4. Repeat for Each integer i from 0 to n:
   - convert i into its binary representation using bin(i).
   - Count the number of 1s in binary string using Count('1').
   - Append this Count to the list ans.
5. Display the list ans as the result.
6. End the program.

Example: Enter a non-negative integer: 5
Count of 1's for numbers from 0 to 5: [0, 1, 1, 2, 1, 2]

Result: Thus the list of length n+1 where each element contains the Count of 1s in the binary representation of its index

Program:

```
def CountBits(n):
    ans = []
    for i in range(n+1):
        ans.append(bin(i).count('1'))
    return ans
# Get input from user
n = int(input("Enter a non-negative integer: "))
result = CountBits(n)
Print("Count of 1's for numbers from 0 to", n, ":", result)
```

4.3

You are given an integer tuple nums containing distinct numbers. your task is to perform a sequence of operations on this tuple until it becomes empty.

AIM: To simulate the given operations on a tuple of distinct integers and count the number of steps needed until the tuple is empty.

Algorithm:

1. start the program
2. Read the tuple nums from the user
3. intialize a counter operations = 0.
4. while nums is not Empty:
    - if nums[0] is the minimum element in nums:
        - Remove it (nums = nums[1:]).
    - Else:
        - move nums[0] to the end of tuple.
        - increment operations by 1.
5. output the value of operations

Example: Enter distinct integers seperated by space: 3 1 2

Number of operations: 5

Program :

```
def Count_operations(nums):
    nums = list(nums) # Convert tuple to list for easier
                      modification
    operations = 0
    while nums:
    if nums[0] == min(nums):
        num.pop(0) # Remove first element
    else:
        nums.appperd(nums.pop()) # move first element to en
    operations += 1
    return operations
# Get input from user
nums = tuple(map(int, input("Enter distinct integer seperated by
                            space:").split()))
Printf("Number of operations:", Count_operation(nums))
```

Task 4.4 :

AIM: To find the single repeated integer in a list where integers are in the range: [1, n] and the list has n+1 elements, using a set for efficient lookup.

Algorithm.

1. start the program.
2. Read the list nums from the user
3. Create an Empty set seen.
4. Iterate through each element num in nums:
   - If num is already in seen, return num as the duplicate
   - otherwise, add num to seen.
5. End.

Example:

Enter numbers separated by space: 3 1 3 4 2

The repeated number is: 3

Result: Thus the, single repeated integer in a list where integers are in the range [1,n] and the list has n+1 elements, using a set for Efficient lookup. has been Executed Successfully.

Program:

```
def find_duplicate(nums):
    seen = set()
    for num in nums:
        if num in seen:
            return num
        seen.add(num)
# Get input from user
nums = list(map(int, input("Enter numbers separated by space: ").split()))
duplicate = find_duplicate(nums)
print("The repeated number is:", duplicate)
```

Task 4.5:

AIM: To write a python Program that stores student details in a dictionary with the student name as the key and their test mark, assignment mark, and lab mark as values. The program will:

1. identify the students(s) with the highest average score.
2. identify the students(s) with the highest assignment marks.
3. identify the student(s) with the lowest lab marks.
4. identify the students with the lowest ~~lab ma~~ score average ~~marks~~.

Algorithm

1. start
2. Read the no. of students n
3. initialize an Empty dictionary students.
4. For Each student from 1 to n:
   - Read the student's name.
   - Read the test mark, assignment mark, and lab mark.
   - store the values as a list in the dictionary with the student name as the key.
5. Define a function average (marks) that returns the average of the given list of marks.
6. Find the highest average score:
   - calculate the average for each student:
   - Find the maximum average (max_avg).

Program:

```
students = {}
n = int(input("Enter number of students: "))
for _ in range():
    name = input("\n Enter student name:")
    test = float(input("Enter Test mark:"))
    assignment = float(input("Enter Assignment mark:"))
        lab = float(input("Enter lab mark:"))
        students[name] = [test, assignment, lab]
# Function to calculate average
    def average(marks):
        return sum(marks) / len(marks)
# 1. Highest average score
    max_avg = max(average(marks) for marks in students.
    values())
highest_avg_students = [name for name, marks in
students.items() if average(marks) == max_avg]
# 2. highest assignment marks
    max_assignment = max(marks[1] for marks in student.values()
    highest_assignment_students = [name for name, marks in
    students.items() if marks[1] == max_assignment]
```

- create a list of all students whose average Equals max_avg.

7. Find the highest assignment marks:
   - Get the assignment mark
   - Find the maximum assignment mark
   - create a list of all students. whose assignment mark Equals max_assignments
8. Find the lowest lab marks:
   - Get the lab mark
   - Find the minimum lab mark
   - create a list of all students whose lab mark Equals min_lab..
9. Find the lowest average score
   - calculate the average for Each student.
   - Find the minimum average (min_avg).
   - create a list of all students whose average Equals min_avg.
10. Display:
    - Highset average score and corresponding student(s).
    - Highest assignment marks and corresponding students(s).
    - lowest lab marks and corresponding student(s).
    - lowest average score and corresponding students(s).
11. stop

#4. Lowest average score

min_avg = min (average (marks) for marks in student.values())

lowest_avg_students = [name for name, marks in students.items() if average (marks) == min_avg]

#Display results

Print ("\n -- Results --")

Print ("Highest average score", highest_avg_students, "with average =", max_avg)

Print ("Highest assignment marks.", highest_assignment_students, "with marks =", max_assignment)

Print ("lowest lab marks", lowest_lab_students, "with marks =", min_lab)

Print ("lowest average score:", lowest_avg_students, "with average =", min_avg)

Example

Enter number of students: 3

Enter student name: Alice

Enter Test mark: 85

Enter Assignment mark: 90

Enter Lab mark: 80

Enter student name: Bob

Enter test mark: 78

Enter Assignment mark: 92

Enter lab mark: 70

Enter Student name: Charlie

Enter Test mark: 85

Enter Assignment mark: 85

Enter lab mark: 85

VEL TECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Output:

Highest average score ['Charlie'] with average = 85.0

Highest assignment marks: ['Bob'] with marks: 92.0

lowest lab marks: ['Bob'] with marks = 70.0

lowest average score: ['Bob'] with average = 80.0

Result: Thus, the program of student dictionary with marks has been Executed successfully.