Task : 4

Date : 30/8/25

USE VARIOUS DATA TYPES, LIST, TUPLES AND DICTIONARY IN PYTHON PROGRAMMING

Aim :-

To write a python program that takes a sorted matrix as input from the user and finds the smallest element that is present in all rows of the matrix. If no such element exists, the program should return -1.

Algorithm :-

1. Start
2. Input the number of rows and columns from the user
3. Initialize an empty matrix mat
4. For each row from 0 to rows - 1:
   - Read the row value from the user as space-separated integers.
   - Append the row to mat.
5. Define a function smallestCommonElement(mat):
   - Assume found = True
   - For each other rows:
     - Perform binary search for num
     - Set left = 0, right = cols - 1.
     - while left <= right
     - if binary search finishes without finding the element, set found = False and break
6. Call the function with mat as input.

Program:-

```
def SmallestCommonElement(mat):
    rows = len(mat)
    cols = len(mat[0])
    for num in mat[0]:
        found = True
        for r in range(1, rows):
            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:
                found = False
                break
        if found:
            return num
    return -1

rows = int(input("Enter number of rows:"))
cols = int(input("Enter number of columns:"))
mat = []
```

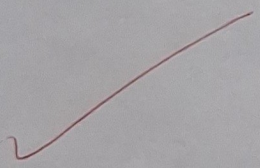
7. Print the result
8. Stop

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



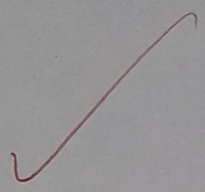
RESULT:-

Thus the Python program has been compiled and executed succesfully.

```
Print("Enter matrix value row by row:")
for _ in range:
    row = list(map(int, input().split()))
    mat.append(row)

result = Smallest Common Element(mat)

Print("Smallest Common Element in all rows:", result)
```



TASK : 4.2

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. Repeate for each integer i from 0 to n:
   - Convert i into its binary representation using bin(i)
   - Count the number of 1s in the 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]

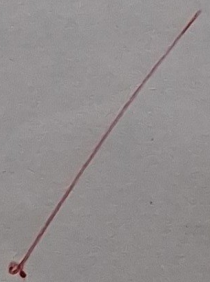
RESULT:-

Thus the Python program has been compiled and executed succesfully.

Program:-

```
def countBits(n):
    ans = []
    for i in range(n+1):
        ans.append(bin(i).count('1'))
    return ans

n = int(input("Enter a non-negative integer: "))
result = countBits(n)
print("Count of 1's for numbers from 0 to "+n+":")
print(result)
```



TASK-4-3:-

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. Initialize 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 the tuple
   - Increment operations by 1
5. Output the value of operations.

EXAMPLE:-

Enter distinct integers separated by space: 3 1 2
Number of operations : 5

RESULT:-

Thus the Python Program has been compiled and executed successfully.

Program :-

```
def count_operations(nums):
    nums = list(nums)
    operations = 0
    while nums:
        if nums[0] == min(nums):
            nums.pop(0)
        else:
            nums.append(nums.pop(0))
            operations += 1
    return operations
nums = tuple(map(int, input("Enter distinct intege
                    separated by space: ").split(
print("Number of operations:", count_operation
```

TASK : 4.4 :-

Aim :-

To find the single repeated integer in a list where integers are in the range of [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 program has been succesfully verified.

PROGRAMS:-

```
def find_duplicate(nums):
    seen = set()
    for num in seen:
        return num
    seen.add(num)
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 test mark, assignment mark, and lab mark as values.

ALGORITHM:-

1. Start
2. Read the number of students n
3. Initialize an empty dictionary students.
4. For each student from 1 to n:
   - Read student's name
   - Read test mark, assignment mark, lab mark
5. Define a function average that returns the average of the given list of marks
6. Find the highest average score:
7. Find the highest assignment marks
8. Find the lowest lab marks
9. Find the lowest average score
10. Display
11. Stop

PROGRAM:-

```
students = {}
n = int(input("Enter number of students: "))
for _ in range(n):
    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]

def average(marks):
    return sum(marks)/len(marks)

#1. Higher 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 Higher assignment marks
max_assignment_students = max(marks[1] for marks in
                              students.values())
highest_assignment_students = [name for name, marks in
                               students.items() if marks[1] == max_assignment]
```

Example :-

```
Enter number of students: 5
Enter student name : Alice
Enter Test marks : 85
Enter Assignment mark : 90
Enter lab mark : 80
Enter Student name: Bob
Enter Test marks : 78
Enter Assignment marks : 92
Enter lab mark : 70
Enter student name : charlie
Enter Test mark : 85
Enter Assignment mark : 85
Enter lab mark : 85
```

O/P :-

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

VELTECH	
EX No.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Result :-

Thus the python program has been compiled & executed successfully.

#3 Lowest lab marks

```
min_lab = min(marks[2] for marks in students.values())
lowest_lab_avg_students = [name for name, marks in
    students.items() if marks[2] == min_lab]
```

#4. Lowest Average Score

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

```
Print("\n -- Result --")
Print("Highest average score:", highest_avg_student
    "with average =", max_avg)
Print("Highest assignment marks:", highest_assign
    -students, "with marks =", max_assign
Print("Lowest Lab marks:", lowest_lab_students,
    "with marks =", min_lab)
Print("Lowest average score:", lowest_average_
    students, "with marks =", min_avg,
```