Experiment 2.1

Student Name: Aditya Patel **UID:** 22BCS11543

Branch: BE-CSE **Section/Group:** 22BCS-IOT-640(B)

Semester: 6th **Date of Performance:** 21/02/25

Subject Name: ADVANCED PROGRAMMING LAB - 2 **Subject Code:** 22CSP-351

PROGRAM-1

1) Aim: Merge Sorted Array.

2) Objective: The objective of the merge function is to combine two sorted arrays into a single sorted array while maintaining the sorted order. This is achieved using a two-pointer technique to efficiently merge elements from both arrays.

3) Implementation/Code:

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n)
      {  for (int j = 0, i = m; j<n; j++){  nums1[i] = nums2[j];  i++;
       }
       sort(nums1.begin(),nums1.end());
    }
};
```

4) Output:

5) Learning Outcomes:

- **Array Manipulation:** Understand how to combine two sorted arrays into one by appending elements and using sorting methods.
- **Time Complexity Awareness:** Recognize the time complexity implications of sorting merged arrays compared to more efficient merging techniques.
- **Edge Case Handling:** Identify and manage scenarios where one or both arrays may be empty.
- **Efficiency Evaluation:** Compare simple implementations with optimized algorithms for merging sorted arrays.
- **Array Manipulation:** Understand how to combine two sorted arrays into one by appending elements and using sorting methods.

PROGRAM-2

1) Aim: Top K Frequent Elements.

2) Objective: The objective of the topKFrequent function is to identify and return the top K most frequent elements from a given vector of integers. The function utilizes a hash map to count the frequency of each element and a priority queue (max heap) to efficiently retrieve the K elements with the highest frequencies.

3) Implementation/Code: class Solution { public:

```
    vector<int> topKFrequent(vector<int>& nums, int k) {
      unordered_map<int, int> ump;
       for(int i: nums){
          ump[i]++;
       }
       priority_queue<pair<int, int>>pq;
      for(auto i: ump){
      pq.push({i.second,i.first});
       }
       vector<int> res;  while(k--){
      auto [elem, count] = pq.top();
      res.push_back(count);
          pq.pop();
       }
       return res;
    }
};
```

4) Output:

```
  if ($?) { g++ exp5.cpp -o exp5 } ; if ($?) { .\exp5 }
Sorted Colors: 0 0 1 1 2 2
Kth Largest Element: 4
PS C:\Users\HP\Desktop\Downloads\ASSIGNMENT AP LAB>
```

5) Learning Outcomes:

- **Understand Frequency Counting:** Learners will grasp how to count the occurrences of elements in a collection using hash maps (unordered maps).
- **Implement Priority Queues:** Participants will learn how to use priority queues (heaps) to efficiently manage and retrieve elements based on their frequency.
- **Develop Problem-Solving Skills:** Learners will enhance their ability to solve algorithmic problems involving frequency analysis and data structure manipulation.
- **Analyze Algorithm Efficiency:** Participants will evaluate the time and space complexity of their solutions, gaining insights into the performance characteristics of their implementations.