Experiment 4

Student Name: Mohit Garg **UID:22BCS50051**

Branch: BE+ME CSE **Section/Group: NTPP_IOT-602(A)**

Semester:6TH **Date of Performance:12/02/25**

Subject Name: AP LAB-2 **Subject Code:22CSP-351**

1. **Aim: Sorting and Searching**
   a) **Merge Sorted Array**
   b) **Top K frequent elements**
   c) **Search a 2D Matrix II**

2. **Code:**

a)
```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        if (n == 0) return;  // If nums2 is empty, no need to merge

        if (m == 0 || nums1[m - 1] <= nums2[n - 1]) {
            nums1[m + n - 1] = nums2[n - 1];
            merge(nums1, m, nums2, n - 1);
        } else {
            nums1[m + n - 1] = nums1[m - 1];
            merge(nums1, m - 1, nums2, n);
        }
    }
};
```

b)
```
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        unordered_map<int, int> counter;
        for (int n : nums) {
            counter[n]++;
        }
```

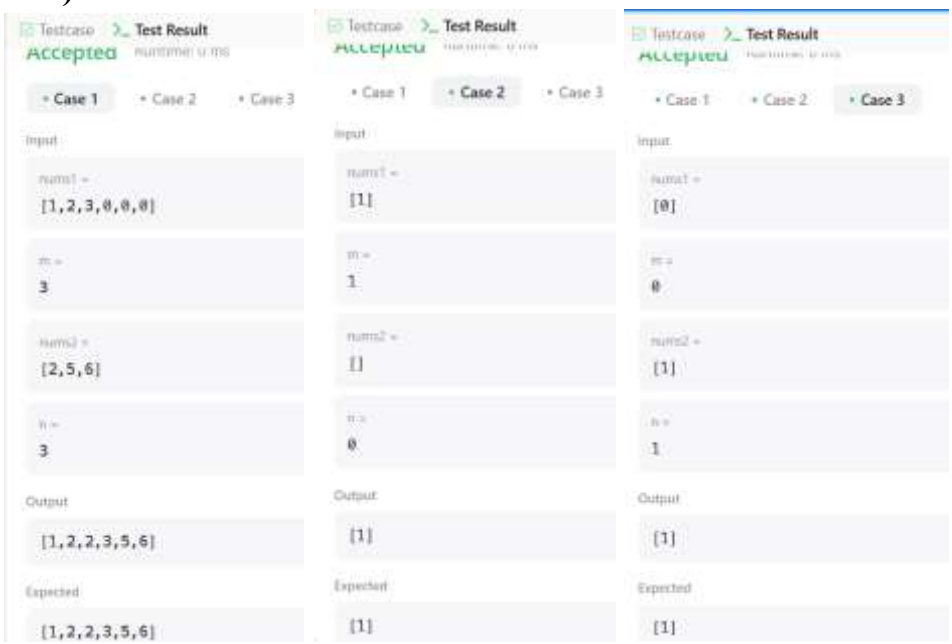
```
        auto comp = [](pair<int, int>& a, pair<int, int>& b) {
            return a.second < b.second;
        };
        priority_queue<pair<int, int>, vector<pair<int, int>>, decltype(comp)> heap(comp);

        for (auto& entry : counter) {
            heap.push({entry.first, entry.second});
        }

        vector<int> res;
        while (k-- > 0) {
            res.push_back(heap.top().first);
            heap.pop();
        }

        return res;
    }
};
```

c)

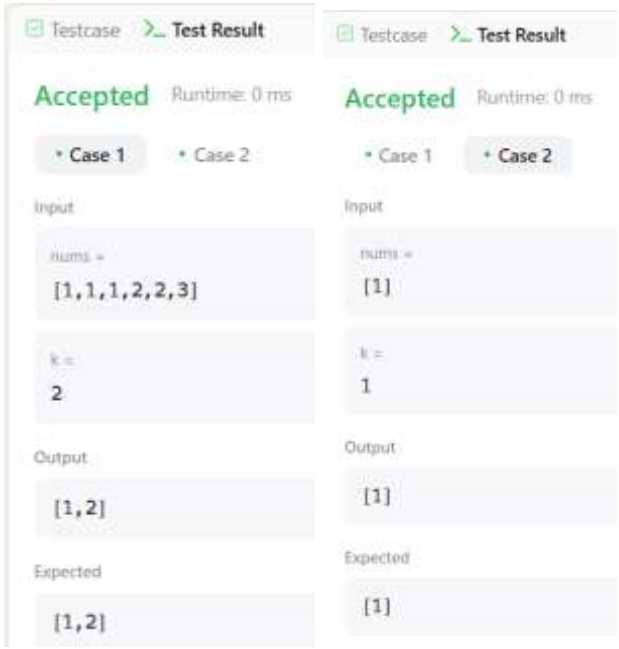
```
public class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        if(matrix == null || matrix.length < 1 || matrix[0].length <1) {
            return false;
        }
        int col = matrix[0].length-1;
        int row = 0;
        while(col >= 0 && row <= matrix.length-1) {
            if(target == matrix[row][col]) {
                return true;
            } else if(target < matrix[row][col]) {
                col--;
            } else if(target > matrix[row][col]) {
                row++;
            }
        }
        return false;
    }
}
```

3. **Output:**

a)

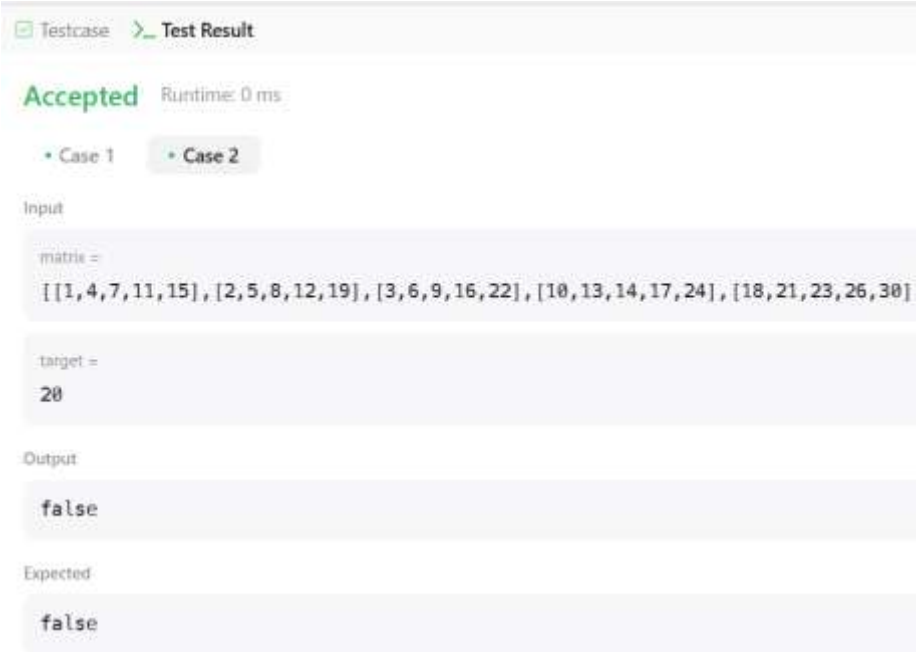

Testcase
Test Result
Accepted
Case 1
Case 2
Case 3
Input
nums1 =
[1,2,3,0,0,0]
m =
3
nums2 =
[2,5,6]
n =
3
Output
[1,2,2,3,5,6]
Expected
[1,2,2,3,5,6]
Testcase
Test Result
Case 1
Case 2
Case 3
Input
nums1 =
[1]
m =
1
nums2 =
[]
n =
0
Output
[1]
Expected
[1]
Testcase
Test Result
Case 1
Case 2
Case 3
Input
nums1 =
[0]
m =
0
nums2 =
[1]
n =
1
Output
[1]
Expected
[1]

b)


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[1,1,1,2,2,3]
k =
2
Output
[1,2]
Expected
[1,2]
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[1]
k =
1
Output
[1]
Expected
[1]

c)


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]
target =
5
Output
true
Expected
true


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]
target =
20
Output
false
Expected
false