Experiment 5

StudentName:- Rishu Sharma **UID:- 22BCS11458**
Branch: CSE **Section/Group:-638-B**
Semester: 6th **Date of Performance- 21/02/2025**
Subject Name: Advanced Programming Lab-2 **Subject Code:** 22CSP-351

Problem -1

1. **AIM:- Kth Largest Element in the Array**

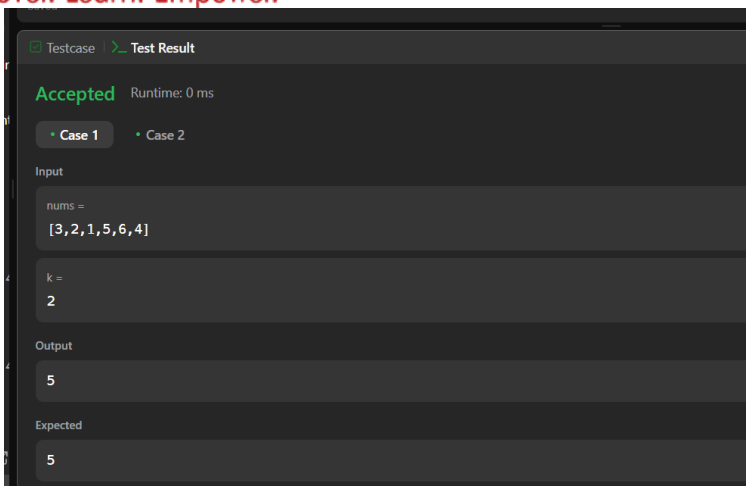
2. **Objective:-**

- Problem Definition: Find the Kth largest element in an unsorted array, meaning the element that ranks Kth when sorted in descending order.
- Sorting Approach (O(n log n)): Sort the array in descending order and directly access the Kth element.
- Heap Approach (O(n log k)): Use a min-heap of size K to keep track of the largest K elements, with the heap's root being the Kth largest.
- Quickselect Algorithm (O(n) on average): A partitioning approach (similar to QuickSort) that finds the Kth largest without sorting the whole array.
- Use Cases: Applied in ranking systems, leaderboards, statistics (percentiles), and real-time data processing where finding top values efficiently is important.
-

3. **CODE:-**

```
class Solution {
        public:
          int findKthLargest(vector<int>& nums, int k) {
            priority_queue<int, vector<int>, greater<int>> minHeap;
            for(int num: nums){
              minHeap.push(num);
              if(minHeap.size() > k){
                minHeap.pop();
              }
            }
          return minHeap.top();
          }
      };
```

2 . Output:-


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[3,2,1,5,6,4]
k =
2
Output
5
Expected
5

Output:


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
nums =
[3,2,3,1,2,4,5,5,6]
k =
4
Output
4
Expected
4

4. Learning Outcome:-

- Understand Different Algorithmic Approaches
- Learn how sorting, heaps, and Quickselect can be used to solve selection problems efficiently.
- Analyze Time and Space Complexity
- Compare the efficiency of different methods
- Implement Efficient Data Structures
- Gain hands-on experience with heaps (priority queues) and partitioning techniques.
- Improve Problem-Solving Skills
- Learn to choose the best algorithm based on constraints (e.g., large vs. small k).
- Apply Concepts in Real-World Scenarios
- Understand its applications in ranking systems, statistics (percentiles), and real-time data processing.

Problem-2

1. Aim: Sort Colors:-

2. Objectives:

- **Sorting Colors Without Sorting Function**: - The goal is to sort an array containing 0s, 1s, and 2s without using built-in sorting. This helps in learning efficient ways to organize data manually.
- **Using the Dutch National Flag Algorithm**: - The algorithm helps in sorting the array in a single pass. This improves understanding of how to arrange elements using multiple pointers.
- **Efficient In-Place Sorting**: - The sorting is done without extra space, modifying the array directly. This teaches how to optimize memory usage in coding problems.
- **Handling Different Cases Easily**: - The method ensures that all numbers are placed in the correct order. It helps in dealing with cases where numbers are shuffled randomly.
- **Improving Logical Thinking and Speed**: - Understanding this approach improves coding skills and speed. It is useful for solving interview questions and competitive programming problems.4

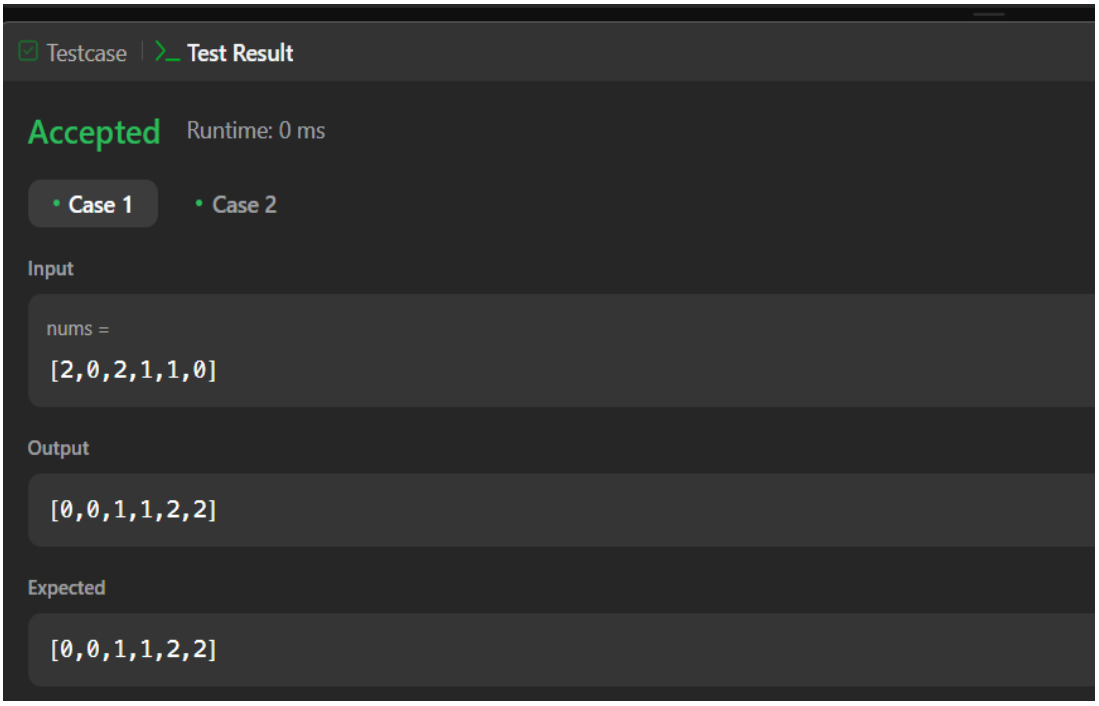
3. Implementation/Code:

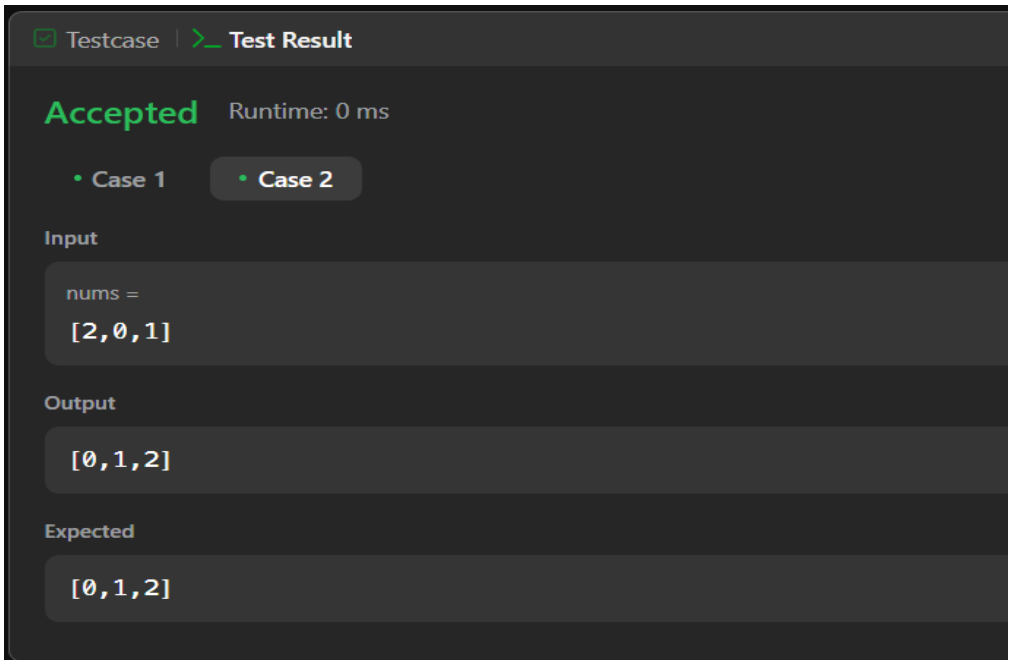
```
 class Solution {
public void sortColors(int[] nums) {
  int left = 0, right = nums.length - 1, current = 0;

  while (current <= right) {
    if (nums[current] == 0) {
      swap(nums, left, current);
      left++;
      current++;
    } else if (nums[current] == 2) {
      swap(nums, right, current);
      right--;
    } else {
      current++;
    }
  }
}

private void swap(int[] nums, int i, int j) {
  int temp = nums[i];
  nums[i] = nums[j];
  nums[j] = temp;
 }
}
```

4. Output:-


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


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

5. Learning Outcomes:-

- Understand the Dutch National Flag Algorithm
- Learn how to sort an array with three distinct values (0, 1, and 2) efficiently.
- Improve Algorithmic Thinking
- Explore different sorting approaches:
- Master Two-Pointer Technique
- Learn how to use low, mid, and high pointers to achieve in-place sorting in a single pass (O(n)).
- Enhance Problem-Solving Skills
- Learn to minimize space complexity and optimize runtime for real-world problems.
- Apply in Real-World Scenarios
- Understand applications in color categorization, bucket sorting, and array partitioning problems.