Experiment 4

Student Name: Ankita
UID: 22BCS17162
Branch: CSE
Section/Group:IOT_NTPP_602-A
Semester: 6th
Date of Performance:10-02-25
Subject Name: AP-2
Subject Code: 22CSP-351

Aim: Sorting and Searching

a) Merge Sorted Array
b) First Bad Version
c) Sort Colors

Objective: To learn and practice Sorting and Searching

Code :

a)

```
#include <vector>
using namespace std;
class Solution {
public:
void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
int i = m - 1;
int j = n - 1;
int k = m + n - 1;
while (i >= 0 && j >= 0) {
if (nums1[i] > nums2[j]) {
nums1[k--] = nums1[i--];
} else {
nums1[k--] = nums2[j--];
}
```

```
}
while (j >= 0) {
nums1[k--] = nums2[j--];
}}};
```

b)

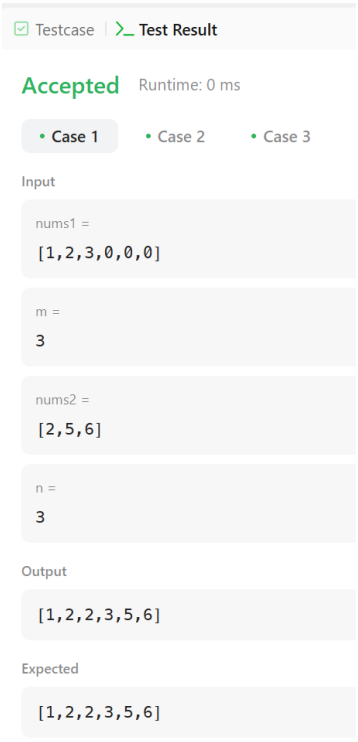
```
class Solution {
public:
int firstBadVersion(int n) {
int left = 1, right = n;
while (left < right) {
int mid = left + (right - left) / 2;
if (isBadVersion(mid)) {
right = mid; // Narrow down to the left half
} else {
left = mid + 1; // Narrow down to the right half
}
}
return left; // Left will be the first bad version
}
};
```

c)

```
#include <vector>
using namespace std;
class Solution {
public:
void sortColors(vector<int>& nums) {
int low = 0, mid = 0, high = nums.size() - 1;
while (mid <= high) {
if (nums[mid] == 0) {
swap(nums[low++], nums[mid++]);
} else if (nums[mid] == 1) {
mid++;
}
else {
swap(nums[mid], nums[high--]);
}
}
}
};
```

Output:

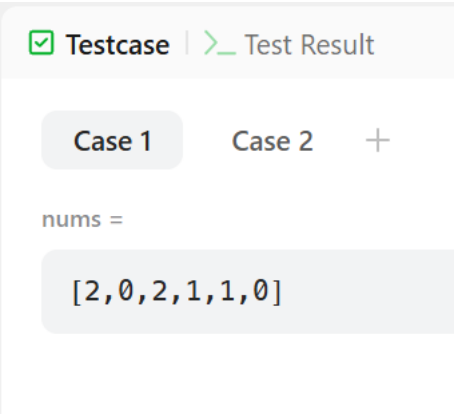
a)


Testcase
Test Result
Accepted
Runtime: 0 ms
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]

b)


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
n =
5
bad =
4
Output
4
Expected
4

c)



Learning Outcomes:

i. Understand how sorting algorithms like Merge Sort and Quick Sort break problems into smaller subproblems and solve them recursively

ii. Learn when to use different sorting/searching techniques based on data size, constraints, and real-time requirements.

iii. Learn how different searching and sorting algorithms perform.