Experiment 5

Student Name: Niraj Kumar **UID: 22BCS16736**
Branch: CSE **Section: 638/A**
Semester: 6th **DOP: 18-02-2014**
Subject: AP **Subject Code:22CSP-351**

Aim:

Problem-1: Beautiful Array

Algorithm:

- **Start with Base Case**
- Begin with a list containing only [1] as the base case.
- **Build Odd and Even Sequences**
- Use a divide-and-conquer approach:
  - Generate odd numbers: 2 * num - 1 (as long as they are ≤ n).
  - Generate even numbers: 2 * num (as long as they are ≤ n).
- Append these numbers in order to ensure no three numbers satisfy 2 * nums[k] == nums[i] + nums[j].
- **Convert List to Array and Return**
- Store the result in an integer array and return it

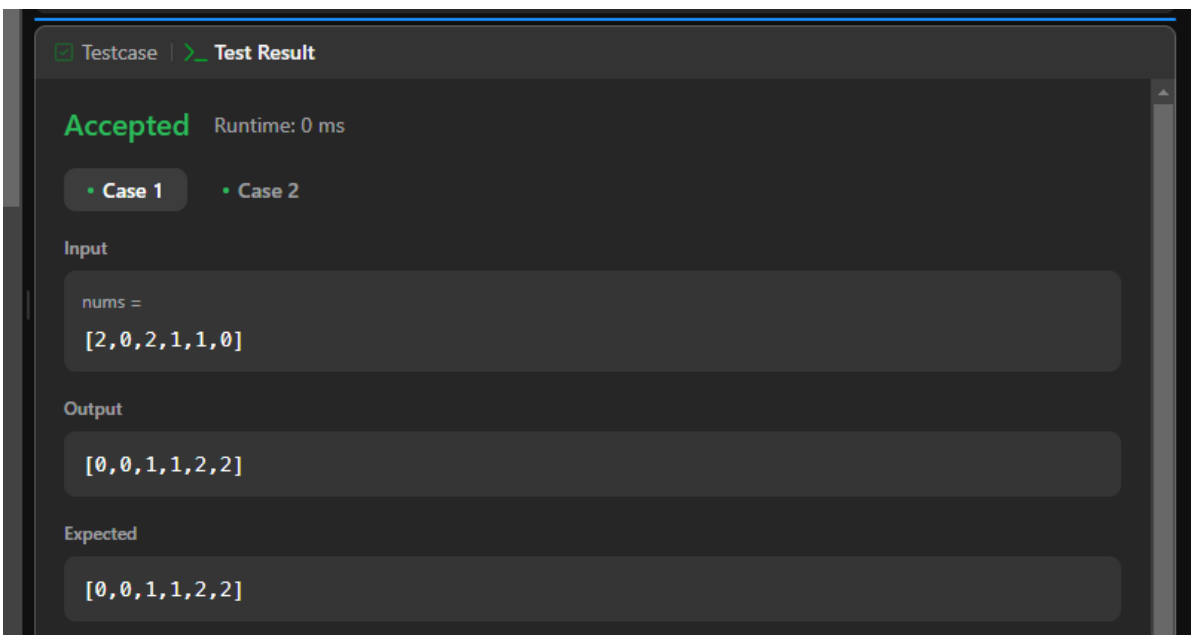
Code:

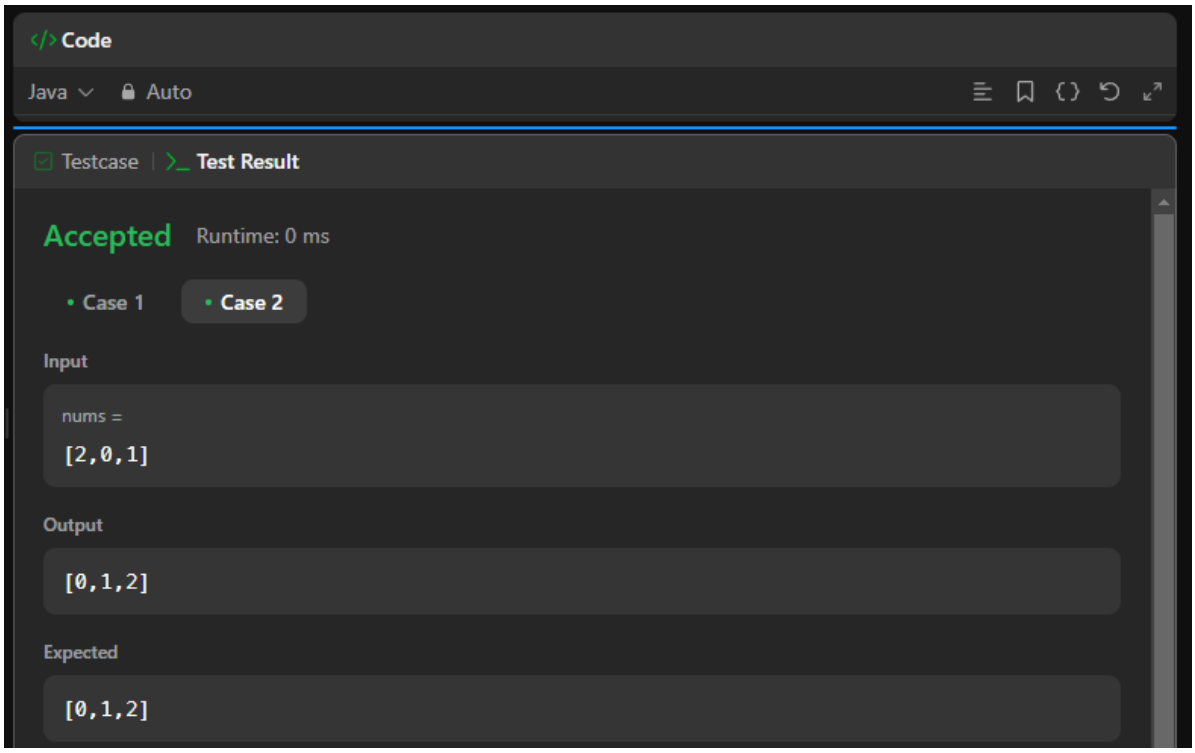
```
import java.util.Arrays;

class Solution {
public void sortColors(int[] nums) {
int low = 0, mid = 0, high = nums.length - 1;
while (mid <= high) {
if (nums[mid] == 0) {
swap(nums, low, mid);
low++;
mid++;
} else if (nums[mid] == 1) {
mid++;
} else {
swap(nums, mid, high);
high--;
}
}
}
```

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

Output:





Aim:

Problem-2: The Skyline Problem

Algorithm :

Use a Min-Heap (Priority Queue)

- Create a min-heap (PriorityQueue) to store the top k largest elements.

Iterate Over the Array

- Insert each number into the min-heap.
- If the heap size exceeds k, remove the smallest element to keep only the top k largest elements.

Return the Kth Largest Element

- The root of the min-heap (peek()) gives the kth largest element in the array.

Code :

```
import java.util.PriorityQueue;
import java.util.*;

class Solution {
public int findKthLargest(int[] nums, int k) {
PriorityQueue<Integer> minHeap = new PriorityQueue<>();
for (int num : nums) {
minHeap.offer(num);
if (minHeap.size() > k) {
minHeap.poll();
}
}
return minHeap.peek();
}


}
```

Output: