Assignment-6

Student Name: Kumar Devashish		**UID**	**:**22BCS10248
Branch	**:** BE-CSE	**Sect./Grp**	**:**FL_IOT-602-A
Semester	**:** 6th	**Date of Performance**	**:**21/02/25
Subject Name	**:** AP Lab	**Subject Code**	**:**22CSP-351

1. **Aim: 108.** Convert Sorted Array to Binary Search Tree

```
class Solution {
    public TreeNode sortedArrayToBST(int[] nums) {
        return buildBST(nums, 0, nums.length - 1);
    }

    private TreeNode buildBST(int[] nums, int left, int right) {
        if (left > right) {
            return null;
        }

        int mid = left + (right - left) / 2;
        TreeNode root = new TreeNode(nums[mid]);
        root.left = buildBST(nums, left, mid - 1);
        root.right = buildBST(nums, mid + 1, right);

        return root;
    }
}
```

2. **Aim:** 191. Number of 1 Bits

```
class Solution {
    public int hammingWeight(int n) {
        int count = 0;
        while (n != 0) {
            count += (n & 1);
            n >>>= 1;
        }
        return count;
    }
}
```

3. **Aim:** 912. Sort an Array

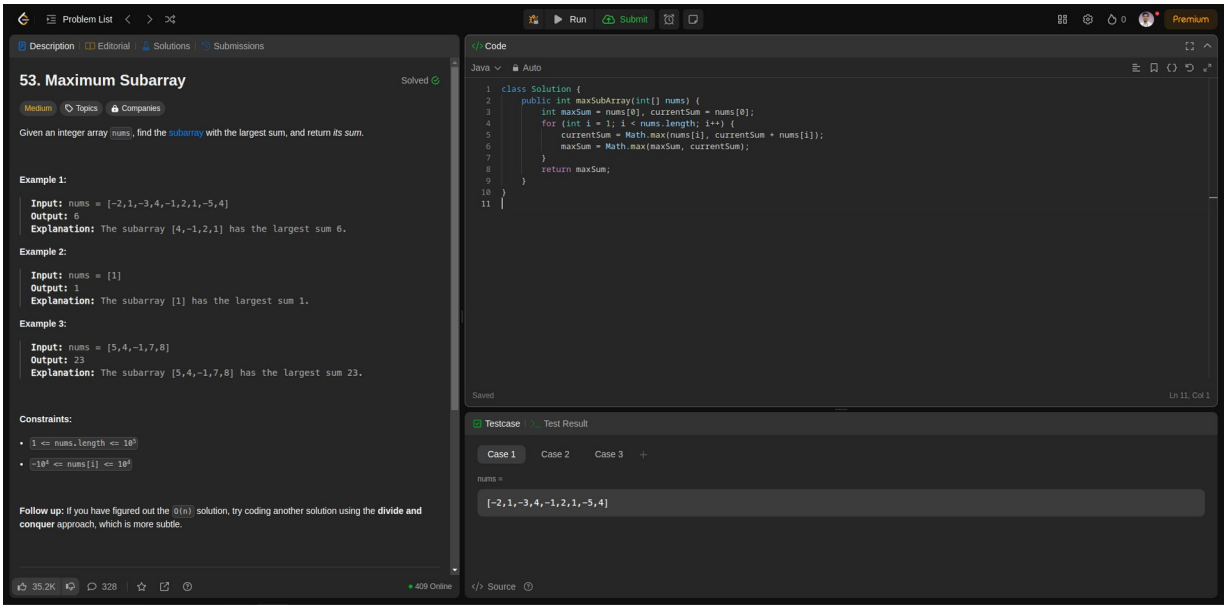
```
import java.util.Arrays;

class Solution {
    public int[] sortArray(int[] nums) {
        Arrays.sort(nums);
        return nums;
    }
}
```

4. **Aim:** 53. Maximum Subarray

```
class Solution {
```

```
    public int maxSubArray(int[] nums) {
        int maxSum = nums[0], currentSum = nums[0];
        for (int i = 1; i < nums.length; i++) {
            currentSum = Math.max(nums[i], currentSum + nums[i]);
            maxSum = Math.max(maxSum, currentSum);
        }
        return maxSum;
    }
}
```

Output:



5. **Aim:** [932. Beautiful Array]

```
import java.util.*;

class Solution {
```

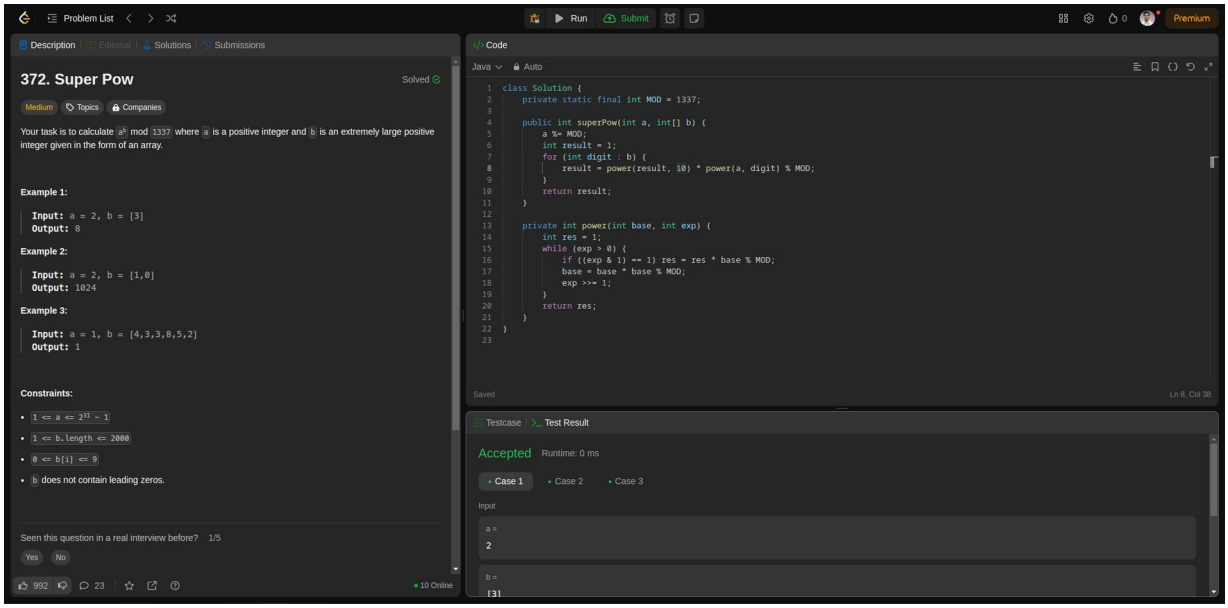
```
    public int[] beautifulArray(int n) {
        List<Integer> result = new ArrayList<>();
        result.add(1);
        while (result.size() < n) {
            List<Integer> temp = new ArrayList<>();
            for (int num : result) {
                if (num * 2 - 1 <= n) temp.add(num * 2 - 1);
            }
            for (int num : result) {
                if (num * 2 <= n) temp.add(num * 2);
            }
            result = temp;
        }
     return result.stream().mapToInt(i -> i).toArray();
    }
}
```

6. Aim: 372. Super Pow

```
class Solution {
    private static final int MOD = 1337;
public int superPow(int a, int[] b) {
        a %= MOD;
        int result = 1;
        for (int digit : b) {
            result = power(result, 10) * power(a, digit) % MOD;
```

```
        }
        return result;
    }
    private int power(int base, int exp) {
        int res = 1;
        while (exp > 0) {
            if ((exp & 1) == 1) res = res * base % MOD;
            base = base * base % MOD;
            exp >>= 1;
        }
        return res;
    }
}
```

Output:



7. Aim: 218. The Skyline Problem

```
import java.util.*;

class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<int[]> events = new ArrayList<>();
        for (int[] b : buildings) {
            events.add(new int[]{b[0], -b[2]});
            events.add(new int[]{b[1], b[2]});
        }

        events.sort((a, b) -> a[0] == b[0] ? Integer.compare(a[1], b[1]) : Integer.compare(a[0],
b[0]));

        List<List<Integer>> result = new ArrayList<>();
        PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
        pq.add(0);
        int prevMax = 0;

        for (int[] e : events) {
            if (e[1] < 0) pq.add(-e[1]);
            else pq.remove(e[1]);

            int currMax = pq.peek();
            if (currMax != prevMax) {
                result.add(Arrays.asList(e[0], currMax));
                prevMax = currMax;
            }
        }

        return result;
    }
}
```