EXPERIMENT - 6

Student Name: JEET BHARTI **UID: 22BCS14804**
Branch: BE-CSE **Section/Group: 602/A**
Semester: 6th **Date of Performance:19-03-25**
Subject Name: Advanced Programming - II **Subject Code: 22CSP-351**

Problems Solved –

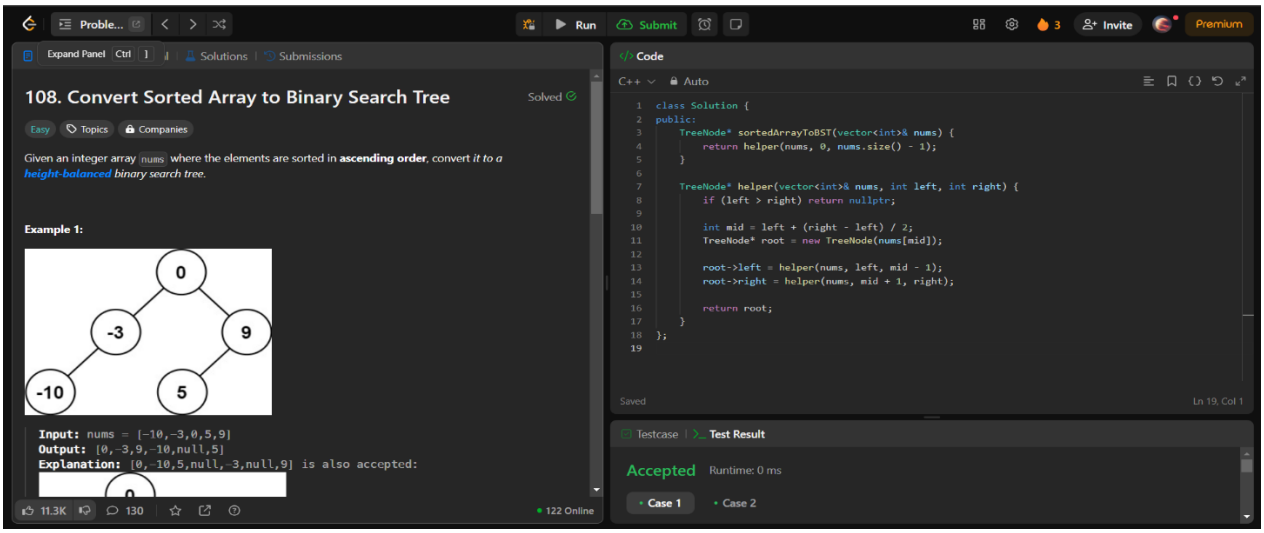
108.Convert Sorted Array to Binary Search Tree
191.Number of 1 Bits
912.Sort an Array
53.Maximum Subarray
932.Beautiful Array
372.Super Pow
218.The Skyline Problem

108. Convert Sorted Array to Binary Search Tree

Aim – Convert a sorted array into a height-balanced binary search tree.

```
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        return helper(nums, 0, nums.size() - 1);
    }
    TreeNode* helper(vector<int>& nums, int left, int right) {
        if (left > right) return nullptr;
        int mid = left + (right - left) / 2;
        TreeNode* root = new TreeNode(nums[mid]);
```

```
        root->left = helper(nums, left, mid - 1);
        root->right = helper(nums, mid + 1, right);
        return root;
    }
};
```

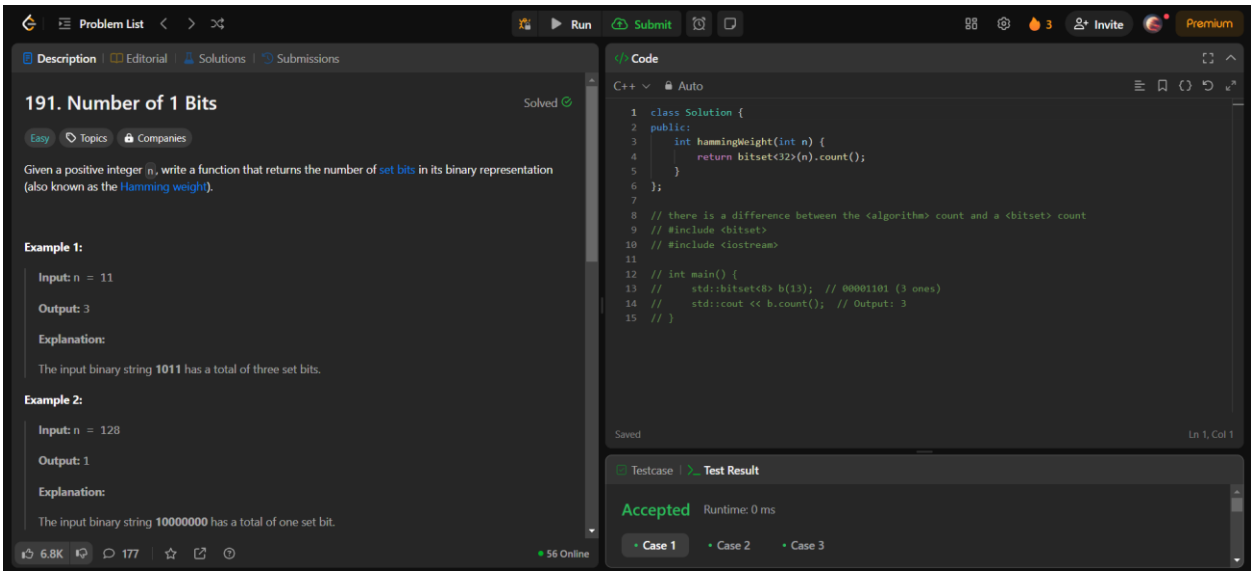


191. Number of 1 Bits

Aim – Count the number of 1 bits in the binary representation of an integer.

CODE:-

```
class Solution {
public:
    int hammingWeight(int n) {
        return bitset<32>(n).count();
    }
};
```

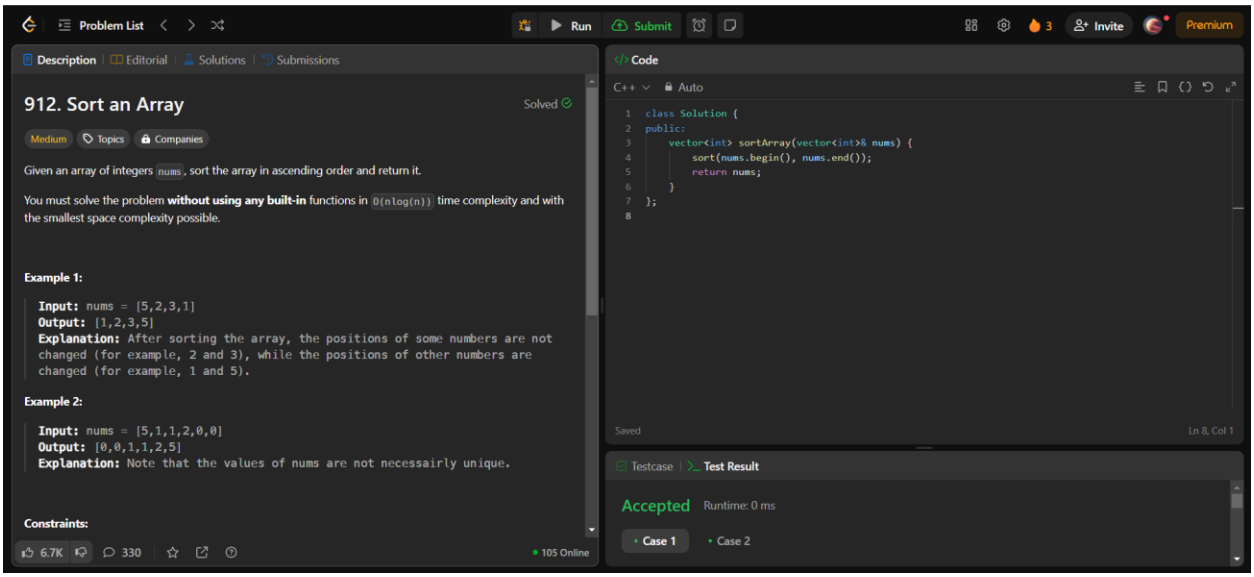


912. Sort an Array

Aim – Sort an array in ascending order.

CODE:-

```
class Solution {
public:
    vector<int> sortArray(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        return nums;
    }
};
```

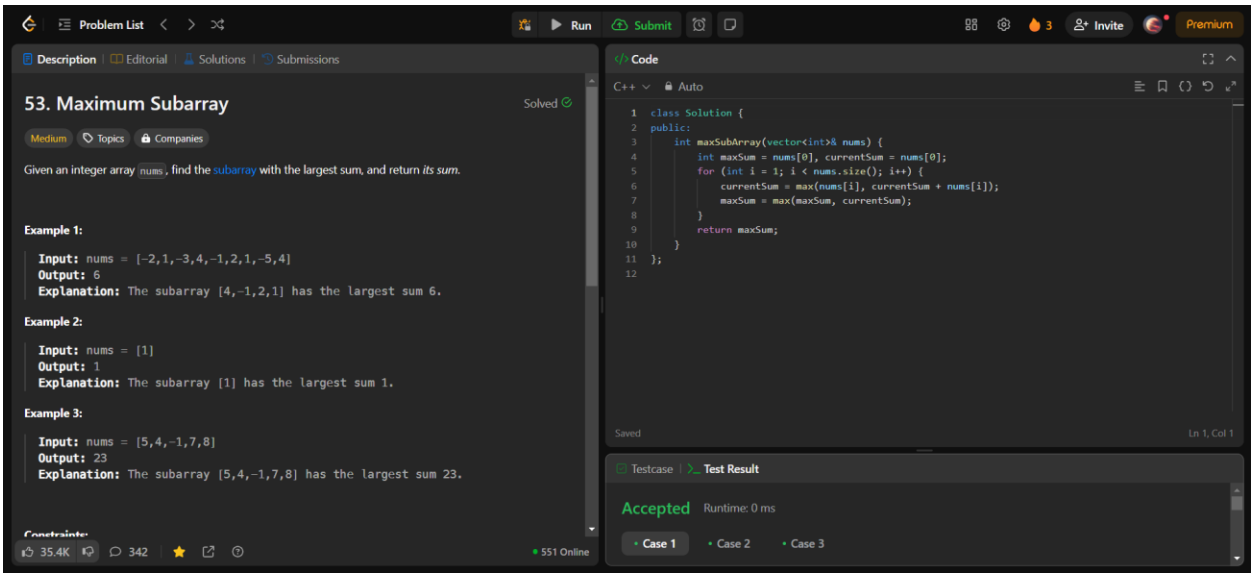


53. Maximum Subarray

Aim – Find the contiguous subarray with the largest sum.

CODE:-

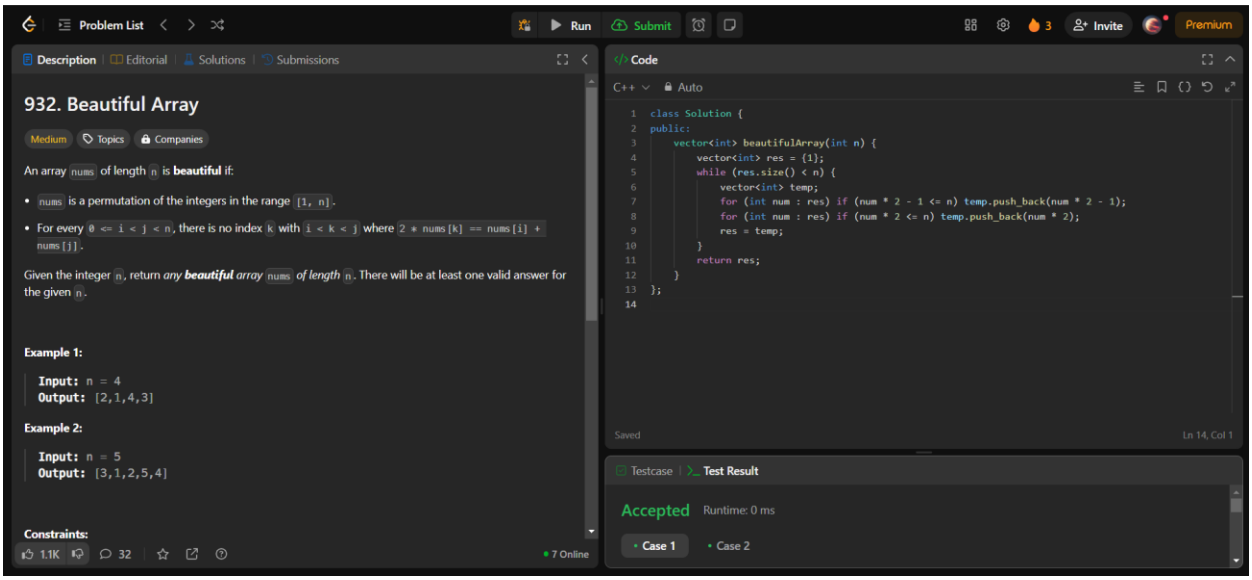
```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0], currentSum = nums[0];
        for (int i = 1; i < nums.size(); i++) {
            currentSum = max(nums[i], currentSum + nums[i]);
            maxSum = max(maxSum, currentSum);
        }
        return maxSum;
    }
};
```



932. Beautiful Array

Aim – Construct a "beautiful" array of integers from 1 to n.

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res = {1};
        while (res.size() < n) {
            vector<int> temp;
            for (int num : res) if (num * 2 - 1 <= n) temp.push_back(num * 2 - 1);
            for (int num : res) if (num * 2 <= n) temp.push_back(num * 2);
            res = temp;
        }
        return res;
    }
};
```



372. Super Pow

Aim – Compute a^b mod 1337, where b is an array of digits.

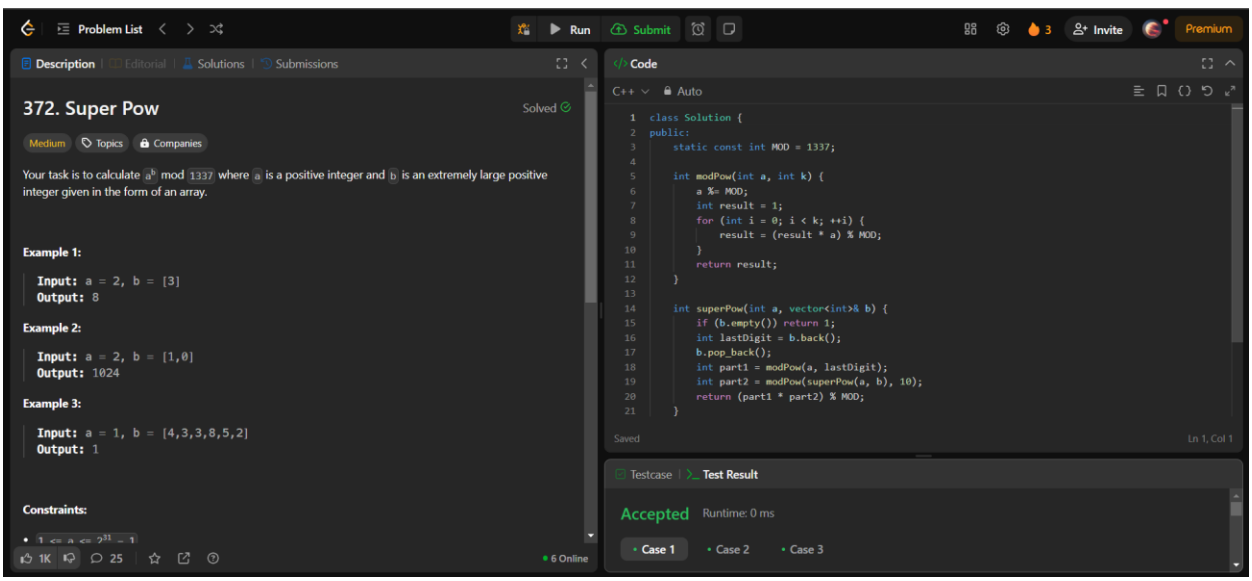
CODE:-

```
class Solution {
public:
    static const int MOD = 1337;

    int modPow(int a, int k) {
        a %= MOD;
        int result = 1;
        for (int i = 0; i < k; ++i) {
            result = (result * a) % MOD;
        }
        return result;
```

```
    }

    int superPow(int a, vector<int>& b) {
        if (b.empty()) return 1;
        int lastDigit = b.back();
        b.pop_back();
        int part1 = modPow(a, lastDigit);
        int part2 = modPow(superPow(a, b), 10);
        return (part1 * part2) % MOD;
    }
};
```



218. The Skyline Problem

Aim – Given a list of buildings, return the key points that form the skyline.

CODE:-

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]);
            events.emplace_back(b[1], b[2]);
        }

        sort(events.begin(), events.end());
        multiset<int> heights = {0};
        vector<vector<int>> result;
        int prevHeight = 0;

        for (auto& e : events) {
            if (e.second < 0) heights.insert(-e.second);
            else heights.erase(heights.find(e.second));

            int currHeight = *heights.rbegin();
            if (currHeight != prevHeight) {
                result.push_back({e.first, currHeight});
                prevHeight = currHeight;
            }
        }
```

```
        return result;
    }
};
```