Experiment 6

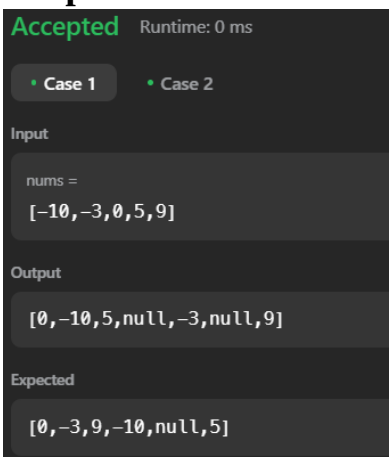
Student Name: Mohd Arshad **UID: 22BCS10091**
Branch: CSE **Section/Group: FL_IOT_602**
Semester: 6TH **Date of Performance: 19/03/25**
Subject Name: AP-II **Subject Code: 22CSP-351**

Q1: Convert Sorted Array to Binary Search Tree

Code:

```
#include <bits/stdc++.h>
using namespace std;
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        return buildBST(nums, 0, nums.size() - 1);
    }
private:
    TreeNode* buildBST(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 = buildBST(nums, left, mid - 1);
        root->right = buildBST(nums, mid + 1, right);
        return root;
    }
};
```

Output:



Q2: Number of 1 Bits

Code:

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

Output:

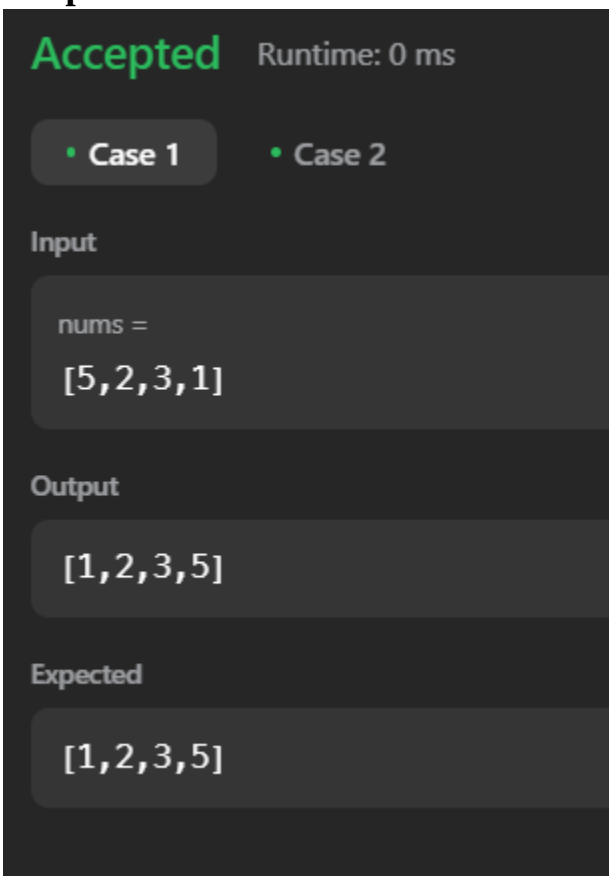
Q3: Sort an Array

Code:

```
#include <bits/stdc++.h>
using namespace std;

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

Output:

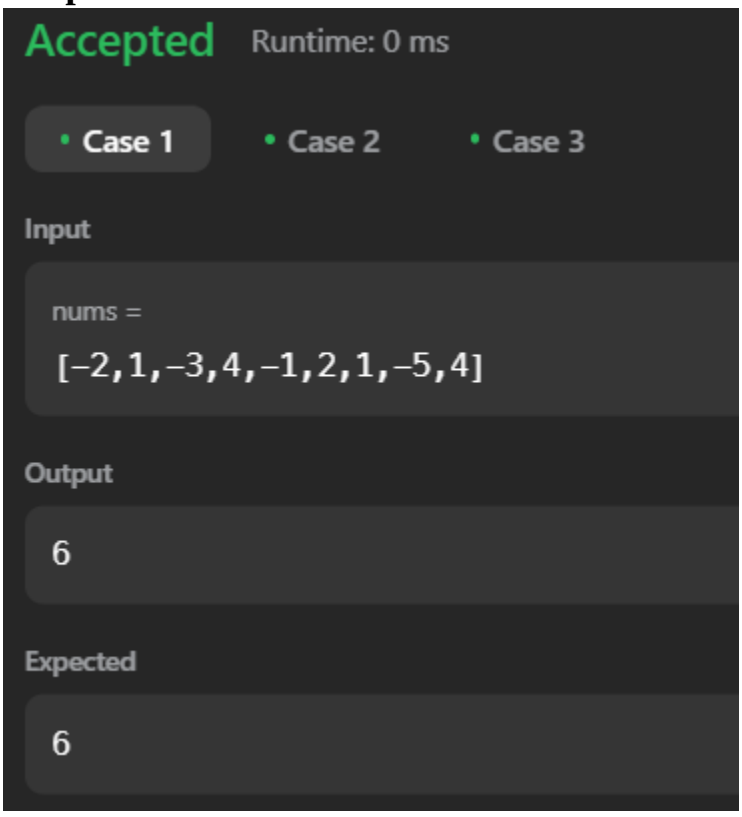


Q4: Maximum Subarray

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;
    }
};
```

Output:

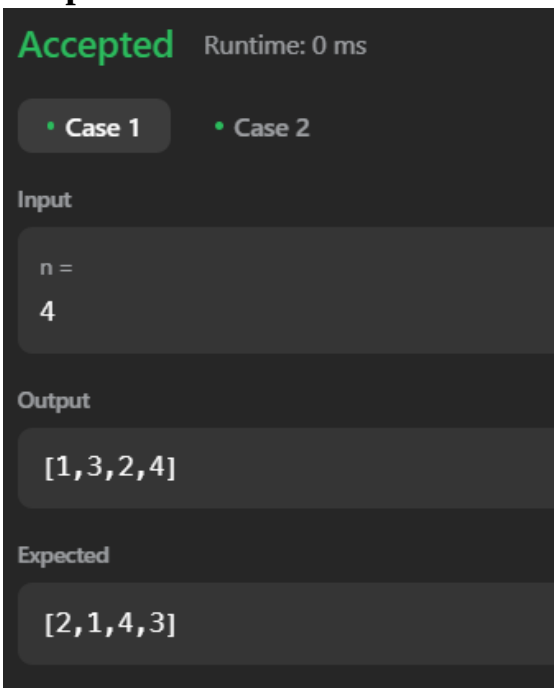


Q5: Beautiful Array

Code:

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

Output:

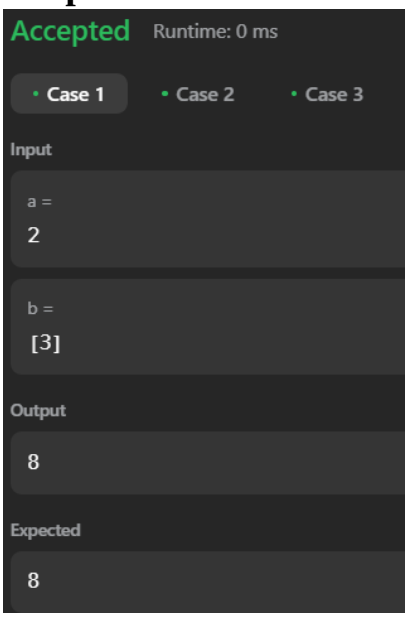


Q6: Super Pow

Code:

```
class Solution {
public:
    int superPow(int a, vector<int>& b) {
        const int MOD = 1337;
        a %= MOD;
        int result = 1;
        for (int i = b.size() - 1; i >= 0; --i) {
            result = (result * powMod(a, b[i], MOD)) % MOD;
            a = powMod(a, 10, MOD);
        }
        return result;
    }
    int powMod(int a, int b, int MOD) {
        int res = 1;
        while (b > 0) {
            if (b % 2 == 1) res = (res * a) % MOD;
            a = (a * a) % MOD;
            b /= 2;
        }
        return res;
    }
};
```

Output:



Q7: The Skyline Problem

Code:

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<vector<int>> result;
        vector<pair<int, int>> heights;
        for (auto& building : buildings) {
            heights.push_back({building[0], -building[2]});
            heights.push_back({building[1], building[2]}); }
        sort(heights.begin(), heights.end());
        multiset<int> activeHeights{0};
        int prevMaxHeight = 0;
        for (auto& height : heights) {
            if (height.second < 0) {
                activeHeights.insert(-height.second);
            } else {
                activeHeights.erase(activeHeights.find(height.second)); }
            int currentMaxHeight = *activeHeights.rbegin();

            if (currentMaxHeight != prevMaxHeight) {
                result.push_back({height.first, currentMaxHeight});
                prevMaxHeight = currentMaxHeight;
            }}
        return result; }
};
```

Output: