Experiment-4(a)

Student Name: Shubham **UID:** 22BCS15490
Branch: BE- CSE **Section/Group:** IOT-637-B
Semester: 6th **Date of Performance:** 13/02/25
Subject Name: AP-II **Subject Code:** 22CSP-359

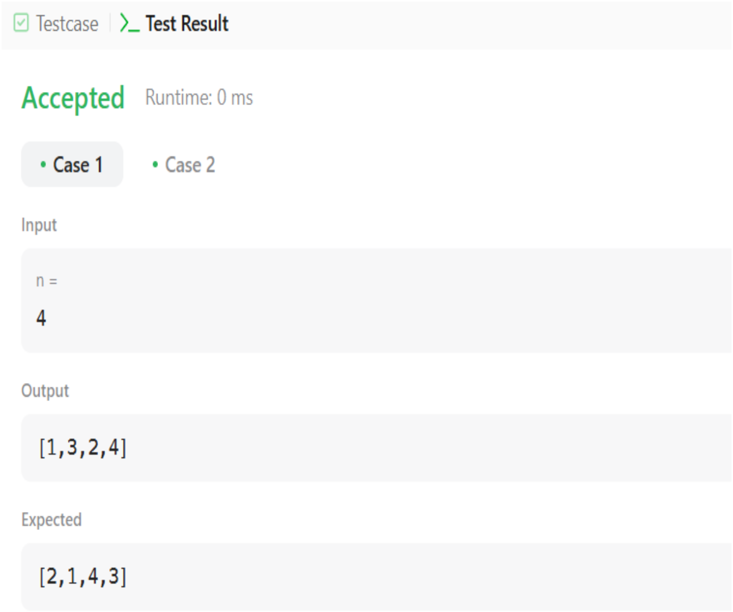
1. **Aim:** To design an algorithm that constructs a beautiful array using an efficient divide-and-conquer approach.

2. **Objective**: Generate a permutation of numbers from 1 to n such that for any pair of indices (i, j), there is no index k between them that satisfies the equation:
   2×nums[k]=nums[i]+nums[j]2 \times nums[k] = nums[i]nums[j]2×nums[k]=nums[i]+nums[j]

3. Code:

```
#include <vector>

using namespace std;

class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};

        vector<int> odd = beautifulArray((n + 1) / 2);  // Recursively build odd part
        vector<int> even = beautifulArray(n / 2);       // Recursively build even part
        vector<int> result;

        // Transform odd elements: 2*x - 1
        for (int x : odd) result.push_back(2 * x - 1);

        // Transform even elements: 2*x
        for (int x : even) result.push_back(2 * x);

        return result;
    }
      };
```

4. Output:



Accepted Runtime: 0 ms

Input

```
n =
4
```

Output

```
[1,3,2,4]
```

Expected

```
[2,1,4,3]
```

5. Learning Outcomes:

- Learn how to break a problem into smaller **subproblems** and combine them efficiently.
- Gain experience in solving problems using **recursion**, especially for combinatorial constraints.
- Learn how to construct sequences using mathematical properties like 2*x - 1 and 2*x.
- Enhance logical thinking and understanding of **permutations and constraints** in algorithm design.

EXPERIMENT-4(b)

1. **AIM:** To develop an efficient algorithm that constructs a skyline representation of a city given a set of buildings.

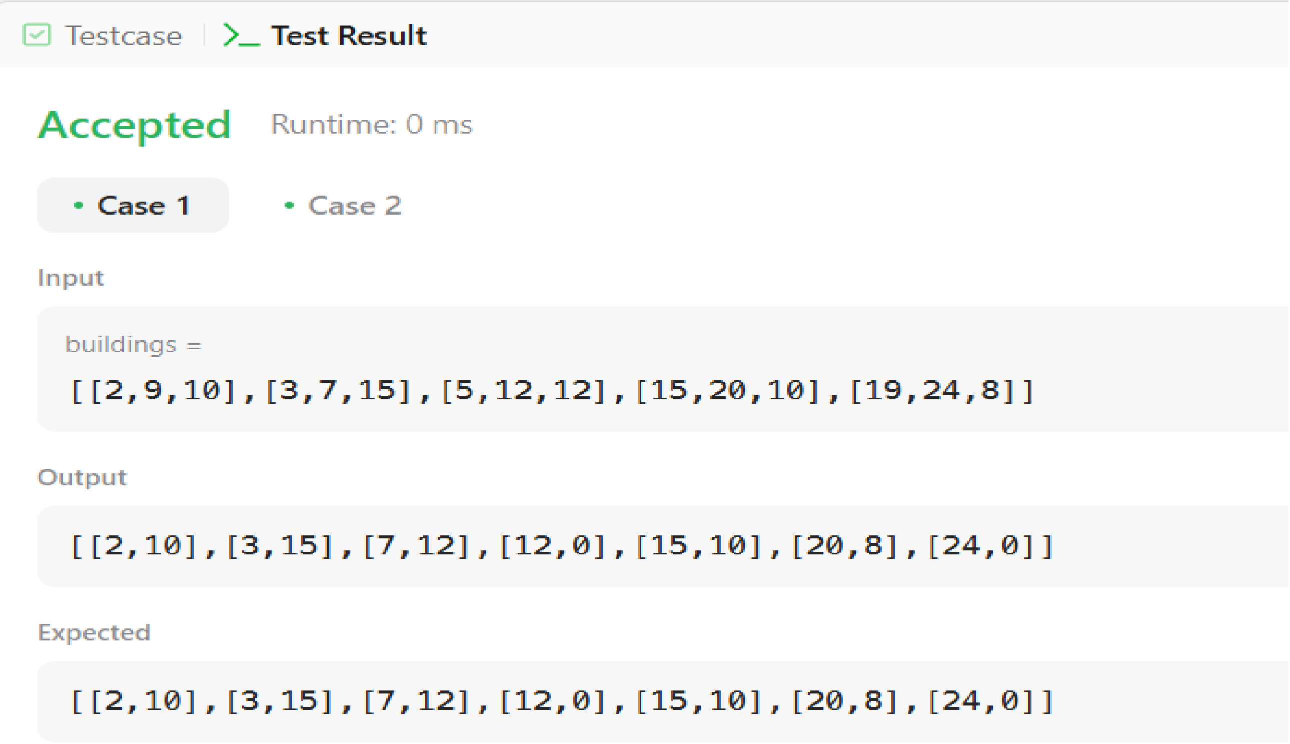
2. **OBJECTIVE**: Understand and implement the Skyline problem by identifying key points in the skyline contour.

3. **CODE:**

```
#include <vector>
#include <set>
#include <algorithm>

using namespace std;

class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        multiset<int> heights = {0};  // Max-Heap using multiset
        vector<vector<int>> result;

        // Step 1: Convert buildings into events (left and right edges)
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]); // Left edge, height negative for insertion
            events.emplace_back(b[1], b[2]);  // Right edge, height positive for removal
        }

        // Step 2: Sort events (first by x, then by height)
        sort(events.begin(), events.end());

        // Step 3: Sweep Line Algorithm
        int prevHeight = 0;
        for (auto& [x, h] : events) {
            if (h < 0) {
                heights.insert(-h); // Insert height (left edge)
            } else {
                heights.erase(heights.find(h)); // Remove height (right edge)
            }

            int currHeight = *heights.rbegin(); // Max height in active buildings
            if (currHeight != prevHeight) { // If height changes, add to result
```

```
                result.push_back({x, currHeight});
                prevHeight = currHeight;
            }
        }
        return result;
    }
};
```

4. **OUTCOME:**



Accepted Runtime: 0 ms

Input

```
buildings =
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```

Output

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

Expected

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

5. **LEARNING OUTCOMES:**

- Learn how to process events in sorted order and track active structures dynamically.
- Gain hands-on experience with multisets, priority queues, and balanced search trees.
- Learn how to maintain a dynamic dataset with fast insertions and deletions.
- Recognize problems that can be solved with divide and conquer or greedy event-based processing.
- Develop optimization strategies to reduce redundant calculations.
- Learn how to process rectangular structures and construct skyline profiles.
- Apply geometric techniques to real-world problems like city planning and CAD modeling.