Experiment 8

Student Name: Gaganjot Singh **UID:** 22BCS13939
Branch: BE-CSE **Section/Group:** 22BCS-JT-802-B
Semester: 5th **Date of Performance:** 14/10/2024
Subject Name: Advanced Programming Lab **Subject Code:** 22CSP-314

1. **Title (1a):** Marc's Cakewalk

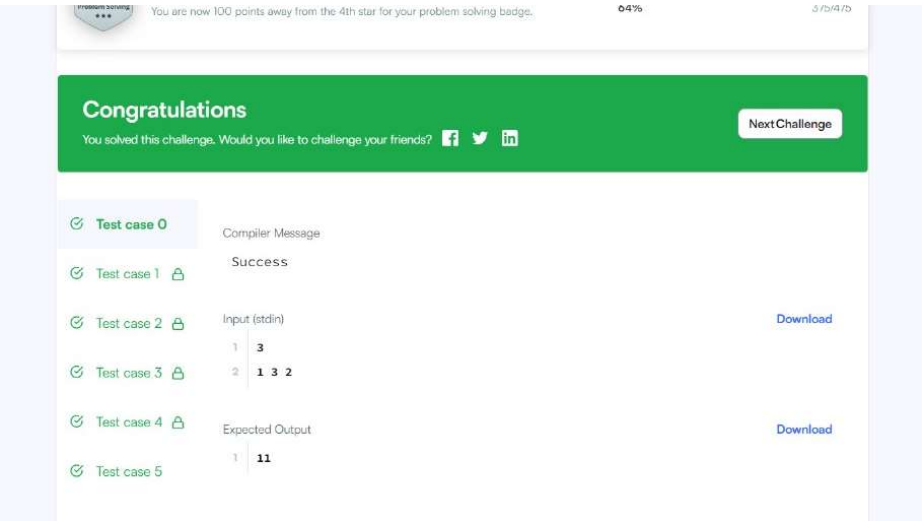
2. **Aim:** Marc loves cupcakes, but he also likes to stay fit. Each cupcake has a calorie count, and Marc can walk a distance to expend those calories. If Marc has eaten j cupcakes so far, after eating a cupcake with ccc calories he must walk at least 2j×c miles to maintain his weight.

3. **Objective:**Print Complete the marcsCakewalk function in the editor below. marcsCakewalk has the following parameters(s):
int calorie[n]: the calorie counts for each cupcake

4. **Code:**

```
long marcsCakewalk(vector<int> calorie) {

    sort(calorie.rbegin(), calorie.rend());

    long miles = 0;


    for (int i = 0; i < calorie.size(); i++) {
        miles += calorie[i] * pow(2, i);
    }

    return miles;
}
```

5. **Output:**



6. **Complexity:**

Time Complexity: O(nlogn)

Space Complexity: O(n)

7. **Learning Outcomes:**
   - Sorting Techniques: Learn to sort vectors in descending order.
   - Using pow Function: Understand calculating powers of numbers.
   - Vector Iteration: Gain experience iterating through vectors.
   - Efficient Calculations: Combine values using exponential scaling.

1. **Title (1b):** Candies

2. **Aim:** Alice is a kindergarten teacher. She wants to give some candies to the children in her class. All the children sit in a line and each of them has a rating score according to his or her performance in the class. Alice wants to give at least 1 candy to each child. If two children sit next to each other, then the one with the higher rating must get more candies. Alice wants to minimize the total number of candies she must buy.

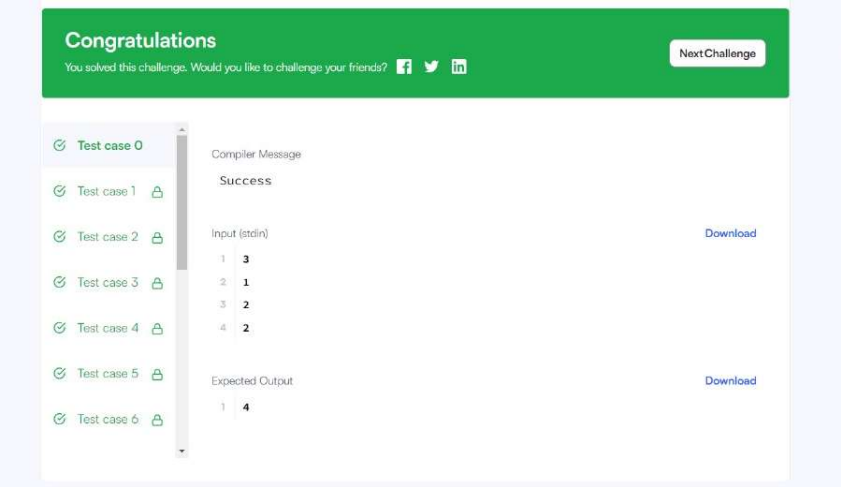
3. **Objective:** Return Complete the candies function in the editor below. candies has the following parameter(s):
   int n: the number of children in the class
   int arr[n]: the ratings of each student

4. **Code:**

```
long candies(int n, vector<int> arr) {

    vector<int> candies(n, 1);

    for (int i = 1; i < n; i++) {
        if (arr[i] > arr[i - 1]) {
            candies[i] = candies[i - 1] + 1;
        }
    }

    for (int i = n - 2; i >= 0; i--) {
        if (arr[i] > arr[i + 1]) {
            candies[i] = max(candies[i], candies[i + 1] + 1);
        }
    }
    long totalCandies = 0;
    for (int i = 0; i < n; i++) {
        totalCandies += candies[i];
    }

    return totalCandies;
```

```
}
```

5. Output:



6. Complexity:

Time Complexity: O(n)

Space Complexity: O(n)

7. Learning Outcomes:

- Understanding Problem Constraints: Learn to address problems with specific constraints (e.g., distributing candies based on ratings).
- Array Initialization: Understand how to initialize an array with default values for tracking state.
- Single Pass Algorithms: Gain experience with two-pass algorithms to efficiently solve problems.
- Comparison Logic: Learn to implement comparison logic to determine values based on conditions.