Experiment 4(A)

Student Name: Tanmaya Kumar Pani **UID: 22BCS12986**
Branch: BE-CSE **Section/Group: IOT-613B**
Semester: 5 **Date of Performance: 13/8/24**
Subject Name: Advanced Programming Lab-1 **Subject Code: 22CSP-314**

1. Title: Missing Numbers

2. Objective:

Given two arrays of integers, find which elements in the second array are missing from the first array.

3. Algorithm:

a) Input: Two vectors, arr and brr.
- arr: A subset of brr with some elements missing.
- brr: The full array with all elements.

b) Sort Both Arrays:
- Sort both arr and brr to compare them easily.

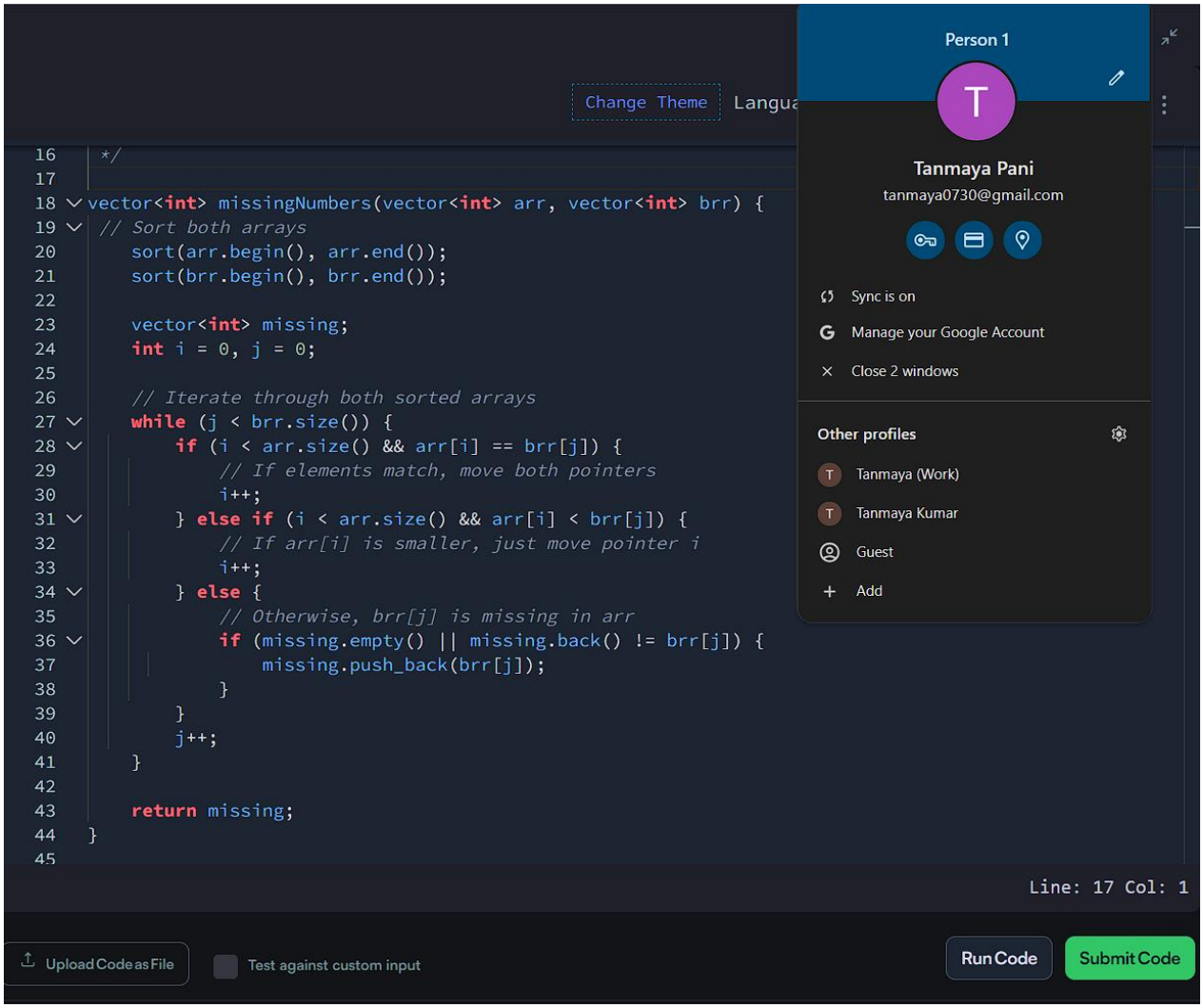
c) Initialize Pointers and Result:
- Create an empty vector missing to store missing numbers.
- Initialize two pointers, i for arr and j for brr.

d) Iterate through Both Arrays:
- Compare elements at arr[i] and brr[j].
  - If arr[i] == brr[j], move both pointers forward.
  - If arr[i] < brr[j], increment pointer i because the current element in arr is smaller and isn't missing.
  - If arr[i] > brr[j], brr[j] is missing in arr. Add brr[j] to the missing vector if it's not a duplicate and move j forward.

e) Continue Until All Elements in brr Are Checked:
- Repeat the comparison and pointer increment until all elements of brr are processed.

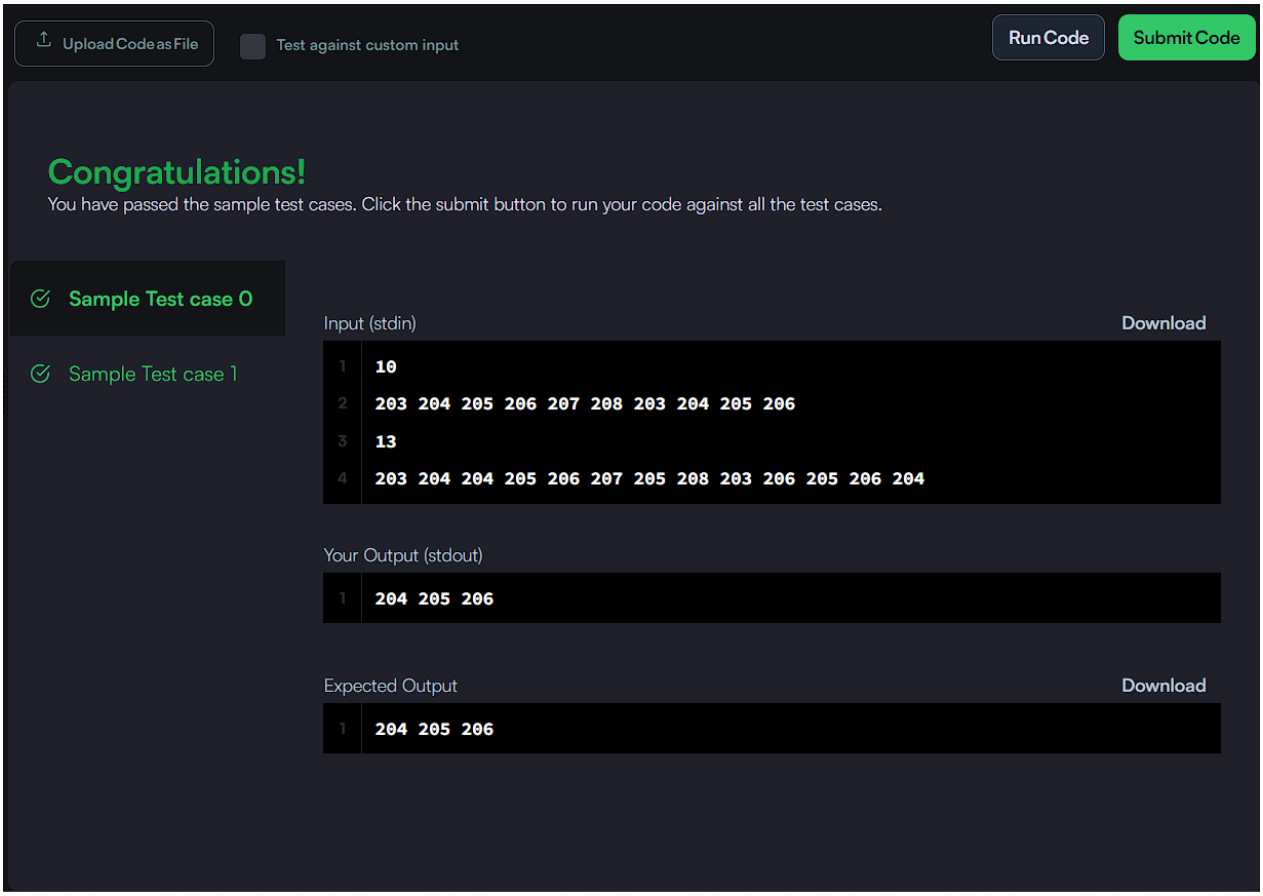
f) Return Result:
- The missing vector contains all the numbers present in brr but missing from arr.

4. Implementation/Code:




```
*/

vector<int> missingNumbers(vector<int> arr, vector<int> brr) {
 // Sort both arrays
    sort(arr.begin(), arr.end());
    sort(brr.begin(), brr.end());

    vector<int> missing;
    int i = 0, j = 0;

    // Iterate through both sorted arrays
    while (j < brr.size()) {
        if (i < arr.size() && arr[i] == brr[j]) {
            // If elements match, move both pointers
            i++;
        } else if (i < arr.size() && arr[i] < brr[j]) {
            // If arr[i] is smaller, just move pointer i
            i++;
        } else {
            // Otherwise, brr[j] is missing in arr
            if (missing.empty() || missing.back() != brr[j]) {
                missing.push_back(brr[j]);
            }
        }
        j++;
    }

    return missing;
}
```

5. Output:



6. Learning Outcomes:

- **Understand Sorting and Comparison**: Learn how sorting helps in efficiently comparing two arrays to find discrepancies, like missing numbers.
- **Efficient Iteration Techniques**: Understand the use of two-pointer techniques to iterate over sorted arrays for optimized performance.
- **Handling Duplicates**: Learn how to avoid adding duplicate missing elements by checking the previous entries.

7. Time Complexity: O(nlogn + mlogm)

8. Space Complexity: O(m)