Date 18/11/2024 day 6 DSA Practice

1. **Bubble Sort**

**Difficulty: EasyAccuracy: 59.33%Submissions: 236K+Points: 2**

**Given an array, arr[]. Sort the array using bubble sort algorithm.**

**Examples :**

**Input: arr[] = [4, 1, 3, 9, 7]**

**Output: [1, 3, 4, 7, 9]**

**Input: arr[] = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]**

**Output: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]**

**Input: arr[] = [1, 2, 3, 4, 5]**

**Output: [1, 2, 3, 4, 5]**

**Explanation: An array that is already sorted should remain unchanged after applying bubble sort.**

**Constraints:**

**1 <= arr.size() <= 103**

**1 <= arr[i] <= 103**

**Code:**

**//{ Driver Code Starts**

**#include <bits/stdc++.h>**

**#include <stdio.h>**

**using namespace std;**

**// } Driver Code Ends**

**// User function Template for C++**

**class Solution {**

**public:**

**// Function to sort the array using bubble sort algorithm.**

**void bubbleSort(vector<int>& arr) {**

**// Your code here**

**for(int i=0;i<arr.size()-1;i++){**

**int swapp=0;**

**for(int j=0;j<arr.size()-i-1;j++){**

**if(arr[j]>arr[j+1]){**

**swap(arr[j],arr[j+1]);**

**swapp=1;**

**}**

**}**

**if(swapp==0){**

**break;**

**}**

**}**

**}**

**};**

**//{ Driver Code Starts.**

**// Driver program to test above functions**

**int main() {**

**int t;**

**cin >> t;**

**cin.ignore();**

**while (t--) {**

**vector<int> arr;**

**string input;**

**getline(cin, input);**

**stringstream ss(input);**

**int number;**

**while (ss >> number) {**

**arr.push\_back(number);**

**}**

**Solution ob;**

**ob.bubbleSort(arr);**

**for (int e : arr) {**

**cout << e << " ";**

**}**

**cout << endl;**

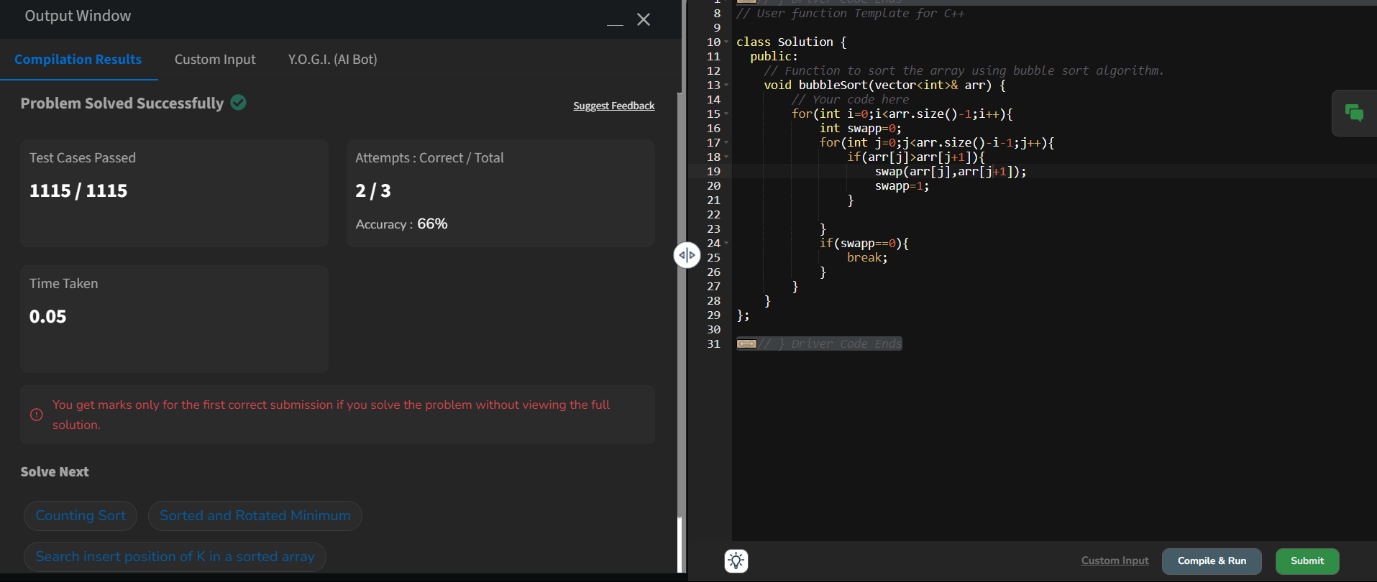
**}**

**return 0;**

**}**

**// } Driver Code Ends**

**Output;**

****

**2.Quick Sort**

Difficulty: **Medium**Accuracy: **55.23%**Submissions: **235K+**Points: **4**

Implement Quick Sort, a Divide and Conquer algorithm, to sort an array, **arr**[] in ascending order. Given an array, **arr**[], with starting index **low** and ending index **high**, complete the functions **partition()** and **quickSort()**. Use the last element as the pivot so that all elements less than or equal to the pivot come before it, and elements greater than the pivot follow it.

**Note**: The **low** and **high** are inclusive.

**Examples:**

**Input:** arr[] = [4, 1, 3, 9, 7]

**Output:** [1, 3, 4, 7, 9]  
**Explanation:** After sorting, all elements are arranged in ascending order.

**Input:** arr[] = [2, 1, 6, 10, 4, 1, 3, 9, 7]

**Output: [**1, 1, 2, 3, 4, 6, 7, 9, 10]  
**Explanation:** Duplicate elements (1) are retained in sorted order.

**Input:** arr[] = [5, 5, 5, 5]

**Output:** [5, 5, 5, 5]  
**Explanation:** All elements are identical, so the array remains unchanged.

**Constraints:**  
1 <= arr.size() <= 103  
1 <= arr[i] <= 104

Code:

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

/\* Function to print an array \*/

void printArray(const vector<int>& arr) {

for (int num : arr)

printf("%d ", num);

printf("\n");

}

// } Driver Code Ends

class Solution {

public:

// Function to sort an array using quick sort algorithm.

void quickSort(vector<int>& arr, int low, int high) {

// code here

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

public:

// Function that takes last element as pivot, places the pivot element at

// its correct position in sorted array, and places all smaller elements

// to left of pivot and all greater elements to right of pivot.

int partition(vector<int>& arr, int low, int high) {

// code here

int pivot = arr[high];

int i = low - 1;

for (int j = low; j < high; j++) {

if (arr[j] < pivot) {

i++;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

return i + 1;

}

};

//{ Driver Code Starts.

int main() {

int T;

scanf("%d", &T);

getchar(); // to consume newline after T

while (T--) {

vector<int> arr;

string input;

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

Solution ob;

ob.quickSort(arr, 0, arr.size() - 1);

printArray(arr);

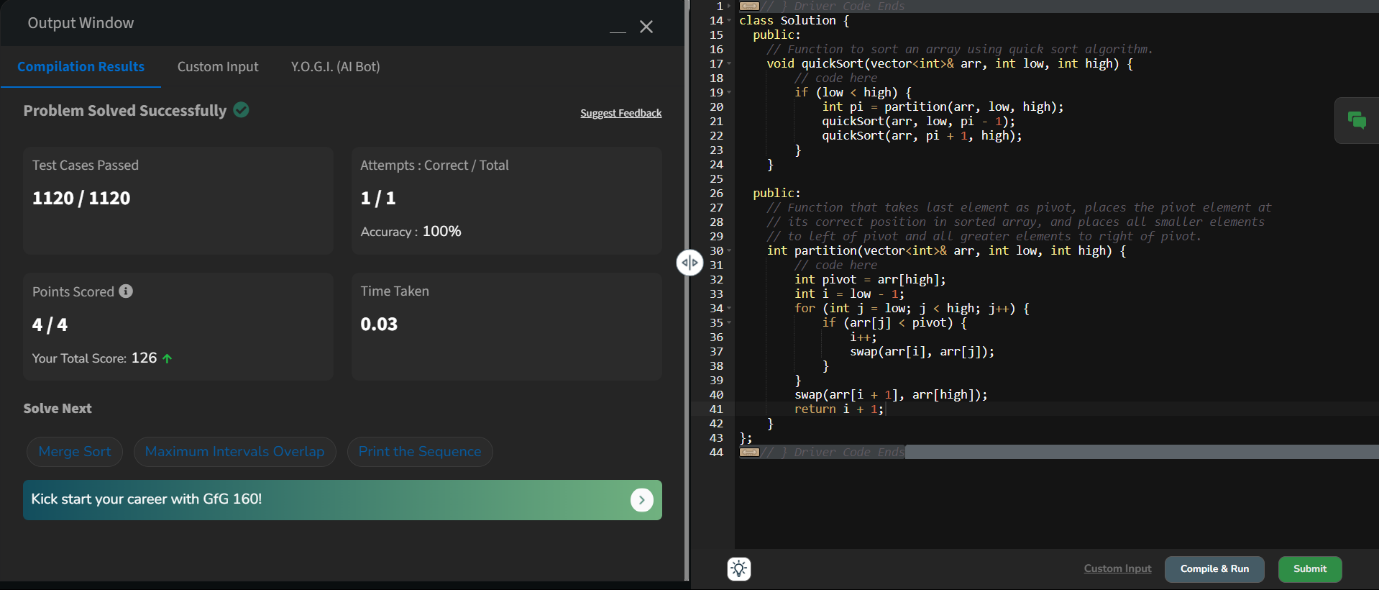
}

return 0;

}

// } Driver Code Ends

Output:



3. **Non Repeating Character**

Difficulty: **Easy**Accuracy: **40.43%**Submissions: **230K+**Points: **2**

Given a string **s** consisting of **lowercase**Latin Letters. Return the first non-repeating character in **s**. If there is no non-repeating character, return **'$'.**  
Note:When you return '$' driver code will output -1.

**Examples:**

**Input:** s = "geeksforgeeks"

**Output:** 'f'

**Explanation:** In the given string, 'f' is the first character in the string which does not repeat.

**Input:** s = "racecar"  
**Output:** 'e'  
**Explanation:** In the given string, 'e' is the only character in the string which does not repeat.

**Input:** s = "aabbccc"  
**Output:** '$'  
**Explanation:** All the characters in the given string are repeating.

**Constraints:**  
1 <= s.size() <= 105

Code:

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

// Function to find the first non-repeating character in a string.

char nonRepeatingChar(string &s) {

// Your code here

map<char,int>m;

for(int i=0;i<s.size();i++){

m[s[i]]+=1;

}

for(int i=0;i<s.size();i++){

if(m[s[i]]==1){

return s[i];

}

}

return '$';

}

};

//{ Driver Code Starts.

int main() {

int T;

cin >> T;

while (T--) {

string S;

cin >> S;

Solution obj;

char ans = obj.nonRepeatingChar(S);

if (ans != '$')

cout << ans;

else

cout << "-1";

cout << endl;

cout << "~"

<< "\n";

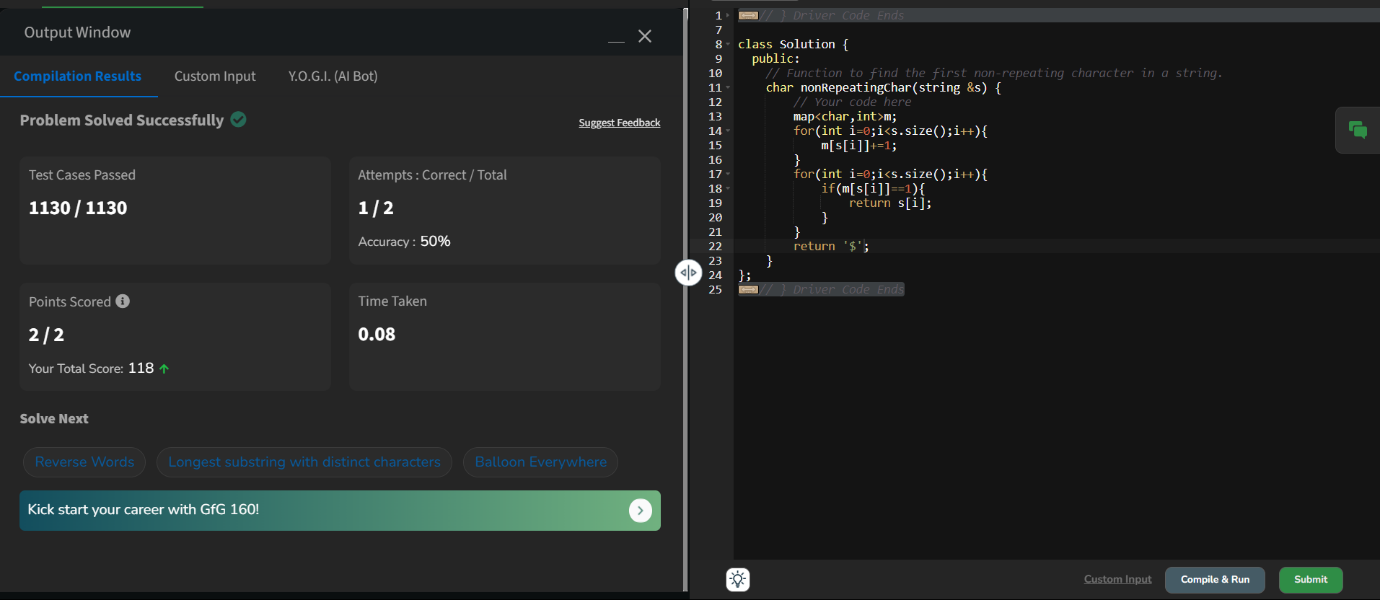
}

return 0;

}

// } Driver Code Ends

Output;



4. **Edit Distance**

Difficulty: **Hard**Accuracy: **35.14%**Submissions: **223K+**Points: **8**

Given two strings **s1** and **s2.**Return the minimum number of operations required to convert **s1**to **s2**.  
The possible operations are permitted:

1. Insert a character at any position of the string.
2. Remove any character from the string.
3. Replace any character from the string with any other character.

**Examples:**

**Input:** s1 = "geek", s2 = "gesek"

**Output:** 1

**Explanation:** One operation is required, inserting 's' between two 'e'.

**Input :** s1 = "gfg", s2 = "gfg"

**Output:** 0

**Explanation:** Both strings are same.

**Input :** s1 = "abc", s2 = "def"

**Output:** 3

**Explanation:** All characters need to be replaced to convert str1 to str2, requiring 3 replacement operations.

**Constraints:**  
1 ≤ s1.length(), s2.length() ≤ 500  
both the strings are in lowercase.

Code:

//{ Driver Code Starts

// Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

// Function to compute the edit distance between two strings

int f(int i,int j,string s1,string s2,vector<vector<int>> &dp){

if(i<0){

return j+1;

}

if(j<0){

return i+1;

}

if(s1[i]==s2[j]){

return dp[i][j]=f(i-1,j-1,s1,s2,dp);

}

if(dp[i][j]!=-1){

return dp[i][j];

}

else{

return dp[i][j]= 1+min(f(i,j-1,s1,s2,dp),min(f(i-1,j,s1,s2,dp),f(i-1,j-1,s1,s2,dp)));

}

}

int editDistance(string s1, string s2) {

int n=s1.size();

int m=s2.size();

vector<vector<int>> dp(n,vector<int>(m,-1));

return f(n-1,m-1,s1,s2,dp);

}

};

//{ Driver Code Starts.

int main() {

int T;

cin >> T;

cin.ignore();

while (T--) {

string s1;

getline(cin, s1);

string s2;

getline(cin, s2);

Solution ob;

int ans = ob.editDistance(s1, s2);

cout << ans << "\n";

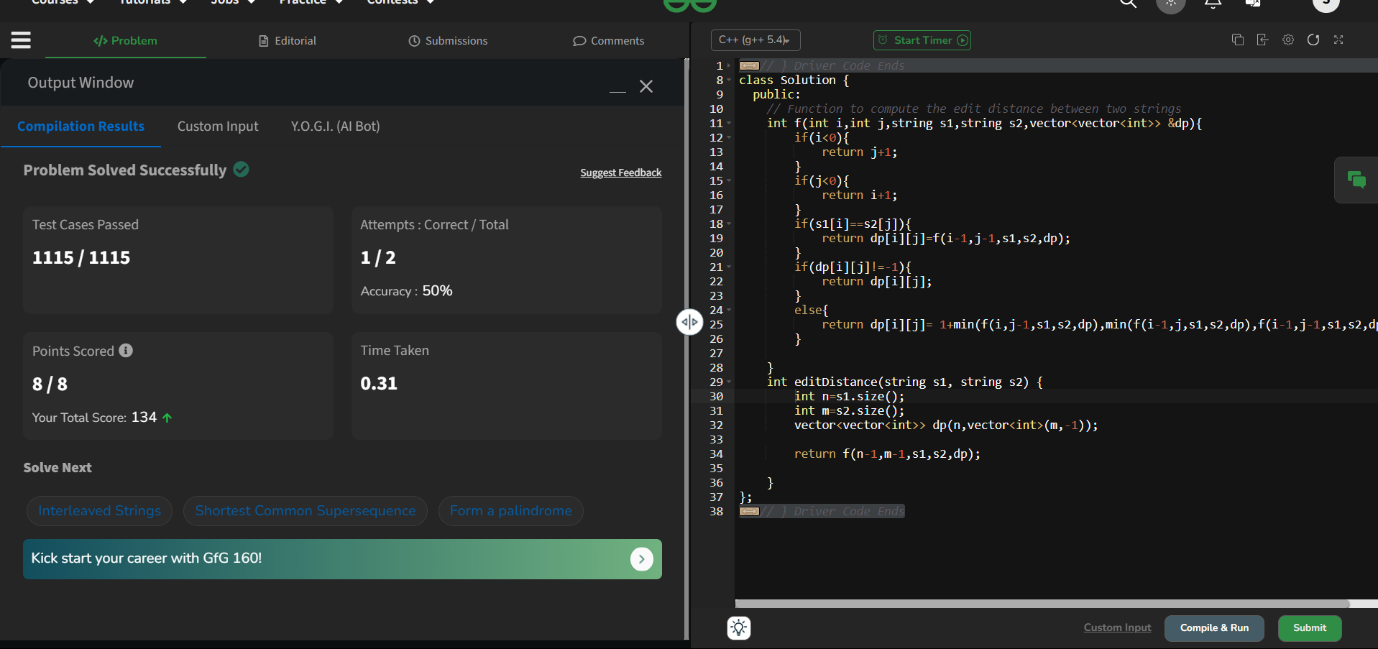
}

return 0;

}

// } Driver Code Ends

Output:



5. **largest elements**

Difficulty: **Medium**Accuracy: **53.56%**Submissions: **163K+**Points: **4**

Given an array **arr[]** of positive integers and an integer **k**, Your task is to return **k largest elements**in decreasing order.

**Examples**

**Input:** arr[] = [12, 5, 787, 1, 23], k = 2

**Output:** [787, 23]

**Explanation:** 1st largest element in the array is 787 and second largest is 23.

**Input:** arr[] = [1, 23, 12, 9, 30, 2, 50], k = 3

**Output:** [50, 30, 23]

**Explanation:** Three Largest elements in the array are 50, 30 and 23.

**Input:** arr[] = [12, 23], k = 1

**Output:** [23]

**Explanation:** 1st Largest element in the array is 23.

**Constraints:**  
1 ≤ k ≤ arr.size() ≤ 106  
1 ≤ arr[i] ≤ 106

Code:

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function template for C++

class Solution {

public:

vector<int> kLargest(vector<int>& arr, int k) {

// code here

priority\_queue<int> p;

vector<int> v;

for(int i=0;i<arr.size();i++){

p.push(arr[i]);

}

for(int i=0;i<k;i++){

v.push\_back(p.top());

p.pop();

}

return v;

}

};

//{ Driver Code Starts.

int main() {

string ts;

getline(cin, ts);

int t = stoi(ts);

while (t--) {

vector<int> arr;

string input;

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

string ks;

getline(cin, ks);

int k = stoi(ks);

Solution ob;

vector<int> ans = ob.kLargest(arr, k);

for (auto it : ans) {

cout << it << " ";

}

cout << endl;

cout << "~" << endl;

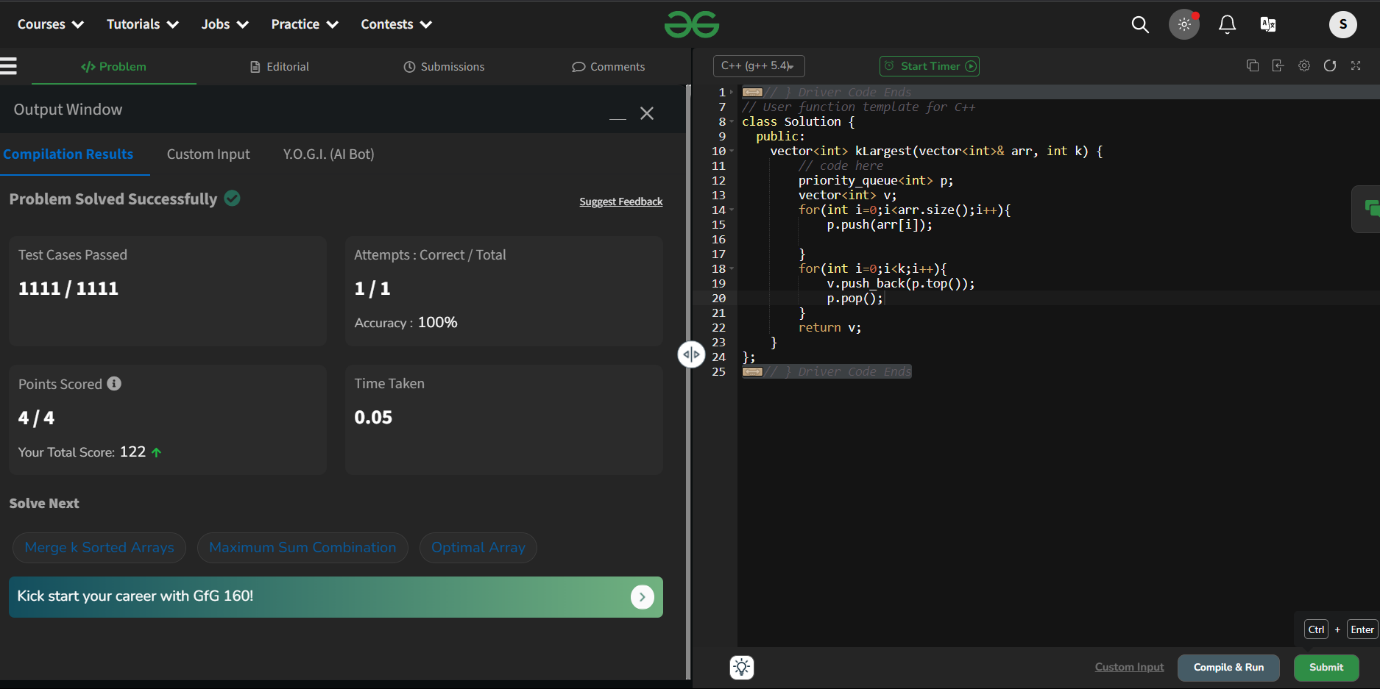
}

return 0;

}

// } Driver Code Ends

Output:



6. **Form the Largest Number**

Difficulty: **Medium**Accuracy: **37.82%**Submissions: **162K+**Points: **4**

Given an array of integers **arr[]**representing non-negative integers, arrange them so that after concatenating all of them in order, it results in the **largest**possible**number**. Since the result may be very large, return it as a string.

**Examples:**

**Input:** arr[] = [3, 30, 34, 5, 9]

**Output:** "9534330"

**Explanation:** Given numbers are {3, 30, 34, 5, 9}, the arrangement "9534330" gives the largest value.

**Input:** arr[] = [54, 546, 548, 60]

**Output:** "6054854654"

**Explanation:** Given numbers are {54, 546, 548, 60}, the arrangement "6054854654" gives the largest value.

**Input:** arr[] = [3, 4, 6, 5, 9]

**Output:** "96543"

**Explanation:** Given numbers are {3, 4, 6, 5, 9}, the arrangement "96543" gives the largest value.

**Constraints:**  
1 ≤ arr.size() ≤ 105  
0 ≤ arr[i] ≤ 105  
The sum of all the elements of the array is greater than 0.

Code:

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// User function template for C++

class Solution {

public:

bool static cmp(string i,string j){

return i+j>j+i;

}

string printLargest(vector<int> &arr) {

// code here

vector<string> arr1;

for(int i=0;i<arr.size();i++){

arr1.push\_back(to\_string(arr[i]));

}

sort(arr1.begin(),arr1.end(),cmp);

string ans = "";

for(int i=0;i<arr1.size();i++){

ans+=arr1[i];

}

return ans;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

cin.ignore();

while (t--) {

vector<int> arr;

string input;

getline(cin, input);

stringstream ss(input);

int number;

while (ss >> number) {

arr.push\_back(number);

}

Solution ob;

string ans = ob.printLargest(arr);

cout << ans << endl;

cout << "~" << endl;

}

return 0;

}

// } Driver Code Ends

Output:

