GE23131-Programming Using C-2024





Input Format

The first line contains an integer, t, denoting the number of trips to the ice cream parlor. The next t sets of lines each describe a visit. Each trip is described as follows:

- 1. The integer **m**, the amount of money they have pooled.
- 2. The integer \mathbf{n}_i the number of flavors offered at the time.
- 3. **n** space-separated integers denoting the cost of each flavor: **cost[cost[1], cost[2],..., cost[n]]**.

Note: The index within the cost array represents the flavor of the ice cream purchased.

Constraints

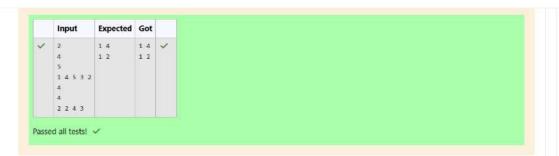
- · 1 ≤ t ≤ 50
- · 2 ≤ m ≤ 10⁴
- . 2 ≤ n ≤ 10⁴
- $1 \le cost[i] \le 10^4$, " i î [1, n]
- · There will always be a unique solution.

Output Format

For each test case, print two space-separated integers denoting the indices of the two flavors purchased, in ascending order.

Sample Input 2 4 5 14 5 3 2 4 4 2 2 4 3 Sample Output 14 12 Explanation Sunny and Johnny make the following two trips to the parlor: 1. The first time, they pool together m = 4 dollars. Of the five flavors available that day, flavors 1 and 4 have a total cost of 1 + 3 = 4. 2. The second time, they pool together m = 4 dollars. TOf the four flavors available that day, flavors 1 and 2 have a total cost of 2 + 2 = 4. Answer: (penalty regime: 0 %)

REC-CIS



Ouestion 2
Correct
Marked out of 5.00
F Flag question

Numeros the Artist had two lists that were permutations of one another. He was very proud. Unfortunately, while transporting them from one exhibition to another, some numbers were lost out of the first list. Can you find the missing numbers?

As an example, the array with some numbers missing, arr = [7, 2, 5, 3, 5, 3]. The original array of numbers brr = [7, 2, 5, 4, 6, 3, 5, 3]. The numbers missing are [4, 6].

Notes

- If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number.
- You have to print all the missing numbers in ascending order.
- Print each missing number once, even if it is missing multiple times.
- The difference between maximum and minimum number in the second list is less than or equal to 100.

Complete the code in the editor below. It should return an array of missing numbers.

It has the following:

- · arr: the array with missing numbers
- · brr: the original array of numbers

Input Format

There will be four lines of input:

n - the size of the first list, arr

The next line contains **n** space-separated integers **arr[i]**

m - the size of the second list, brr

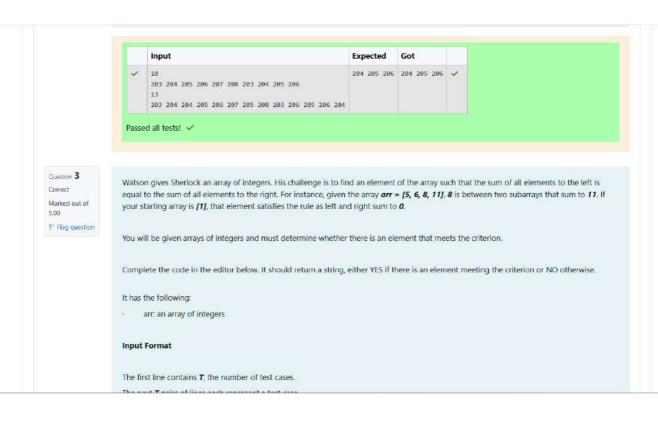
The next line contains **m** space-separated integers **brr[i]**

Constraints

- · 1 ≤ n, m ≤ 2 x 10⁵
- . n ≤ m
- · 1 ≤ brr[i] ≤ 2 x 104
- $X_{max} X_{min} < 107$

Output Format

REC-CIS



- The first line contains n	the number o	f elements in the array arr.
-----------------------------	--------------	------------------------------

- The second line contains n space-separated integers $\mathit{arr[i]}$ where $0 \le i < n$.

Constraints

- · 1 ≤ T ≤ 10
- $1 \le n \le 10^5$
- $1 \leq arr[i] \leq 2 \times 10^4$
- . 0≤i≤n

Output Format

For each test case print YES if there exists an element in the array, such that the sum of the elements on its left is equal to the sum of the elements on its right; otherwise print NO.

Sample Input 0

2

3

123

4

1233

Explanation 0 For the first test case, no such index exists. For the second test case, arr[0] + arr[1] = arr[3], therefore index 2 satisfies the given conditions. Sample Input 1 3 5 11411 4 2000 4 0020 Sample Output 1 YES YES YES Explanation 1

```
REC-CIS
```

```
In the first test case, arr[2] = 4 is between two subarrays summing to 2.
In the second case, arr[0] = 2 is between two subarrays summing to 0.
In the third case, arr[2] = 2 is between two subarrays summing to 0.
Answer: (penalty regime: 0 %)
   1 #include<stdio.h>
    2 int main()
3 * {
              int t,n,lsum,rsum,middle;
scanf("%d",&t);
for(int i=0;i<t;i++)</pre>
                    1sum=0;
                   rsum=0;
scanf("%d",&n);
int arr[n];
for(int j=0;j<n;j++)
   10
11
12
13
14
15
16
17
18
19
                        scanf("%d",&arr[j]);
                   }
middle=n/2;
if(arr[middle]=0){
   for(middle=0;(arr[middle]=0)&&middle<n;middle++);</pre>
                    for(int j=0;j<=middle;j++)</pre>
   20
                         1sum+=arr[j];
   22
23
24
25
26
27
                    for(int j=middle;j<n;j++)</pre>
                         rsum⊹arr[j];
   28
29
                    printf("%s\n",(lsum==rsum)?"YES":"NO");
```

}