Question 1

Correct

Marked out of 0.00

Flag question

Sunny and Johnny like to pool their money and go to the ice cream parlor. Johnny never buys the same flavor that Sunny does. The only other rule they have is that they spend all of their money.

Given a list of prices for the flavors of ice cream, select the two that will cost all of the money they have.

For example, they have $m = 6$ to spend and there are flavors costing $cost = [1, 2, 3, 4, 5, 6]$. The two flavors costing **1** and **5** meet the criteria. Using **1**-based indexing, they are at indices **1** and **4**.

Function Description

Complete the code in the editor below. It should return an array containing the indices of the prices of the two flavors they buy.

It has the following:

m: an integer denoting the amount of money they have to spend.

cost: an integer array denoting the cost of each flavor of ice cream.

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 n, the number of flavors offered at the time.
3. n space-separated integers denoting the cost of each flavor: $cost[cost[1], cost[2], \ldots, cost[n]]$

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

Constraints

- $1 \le t \le 50$
- $2 \le m \le 10^4$
- $2 \le n \le 10^4$
- $1 \le cost[i] \le 10^4, \forall i \in [1, n]$
- 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
1 4 5 3 2
4
4
2 2 4 3
```

Sample Output

```
1 4
1 2
```

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. Of the four flavors available that day, flavors **1** and **2** have a total cost of $2 + 2 = 4$.

Answer: (penalty regime: 0 %)

```
#include<stdio.h>
int main()
{
    int t,m,n,i,j,found;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d %d",&m,&n);
        int cost[n];
        for(i=0;i<n;i++)
        {
            scanf("%d",&cost[i]);
        }
        found = 0;
        for(i=0;i<n-1;i++)
        {
            for(j=i+1;j<n;j++)
            {
                if(cost[i]+cost[j]==m)
                {
                    printf("%d %d\n",i+1,
                    found = 1;
                    break;
                }
            }
            if (found)
            {
                break;
            }
        }
    }
    return 0;
}
```

	Input	Expected	Got	
✓	2 4 5 1 4 5 3 2 4 4 2 2 4 3	1 4 1 2	1 4 1 2	✓

Question 2

Correct

Marked out of 5.00

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

- $1 \leq n, m \leq 2 \times 10^5$
- $n \leq m$
- $1 \leq brr[i] \leq 2 \times 10^4$
- $X_{max} - X_{min} < 101$

Output Format

Output the missing numbers in ascending order.

Sample Input

```
10
203 204 205 206 207 208 203 204 205 206
13
203 204 204 205 206 207 205 208 203 206 205 206 204
```

Sample Output

```
204 205 206
```

Explanation

204 is present in both arrays. Its frequency in ***arr*** is **2**, while its frequency in ***brr*** is **3**. Similarly, **205** and **206** occur twice in ***arr***, but three times in ***brr***. The rest of the numbers have the same frequencies in both lists.

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int n,m,c,co,c1=0;
    scanf("%d",&n);
    int arr[n];
    for(int a=0;a<n;a++)
    {
        scanf("%d",&arr[a]);
    }
    scanf("%d",&m);
    int brr[m],ans[m];
    for(int b=0;b<m;b++)
    {
        scanf("%d",&brr[b]);
    }
    for(int j=0;j<m;j++)
    {
        c=0;
        for(int i=0;i<n;i++)
        {
            if(arr[i]==brr[j])
            {
                c=1;
                arr[i]=-1;
                break;
            }
        }
        if(c==0)
        {
            ans[c1]=brr[j];
            c1++;
        }
    }
    for(int a=0;a<c1;a++)
    {
        co=0;
        for(int b=0;b<c1;b++)
        {
            if(ans[b]<ans[a])
            co++;
        }
        int temp=ans[a];
        ans[a]=ans[co];
        ans[co]=temp;
    }
    for(int i=0;i<c1;i++)
    printf("%d ",ans[i]);
}
```

	Input
✓	10 203 204 205 206 207 208 203 204 205 206 13 203 204 204 205 206 207 205 208 203 206 205

Question 3

Correct

Marked out of 5.00

Flag question

Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right. For instance, given the array ***arr = [5, 6, 8, 11], 8*** is between two subarrays that sum to ***11***. If your starting array is ***[1]***, that element satisfies the rule as left and right sum to ***0***.

You will be given arrays of integers and must determine whether there is an element that meets the criterion.

Complete the code in the editor below. It should return a string, either YES if there is an element meeting the criterion or NO otherwise.

It has the following:

arr: an array of integers

Input Format

The first line contains ***T***, the number of test cases.

The next ***T*** pairs of lines each represent a test case.

- The first line contains ***n***, the number of elements in the array ***arr***.
- The second line contains ***n*** space-separated integers ***arr[i]*** where ***0 ≤ i < n***.

Constraints

- ***1 ≤ T ≤ 10***
- ***1 ≤ n ≤ 10^5***
- ***1 ≤ arr[i] ≤ 2 x 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
1 2 3
4
1 2 3 3
```

Sample Output 0

```
NO
YES
```

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
1 1 4 1 1
4
2 0 0 0
4
0 0 2 0
```

Sample Output 1

```
YES
YES
YES
```

Explanation 1

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 %)

```c
#include<stdio.h>
int main()
{
    int T,n,i,left_sum,right_sum;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        int arr[n];
        for(i=0;i<n;i++)
        {
            scanf("%d",&arr[i]);
        }
        left_sum = 0;
        right_sum = 0;
        for(i=0;i<n;i++)
        {
            right_sum += arr[i];
        }
        for(i=0;i<n;i++)
        {
            right_sum -= arr[i];
            if(left_sum == right_sum)
            {
                printf("YES\n");
                break;
            }
            left_sum += arr[i];
        }
        if(left_sum != right_sum)
        {
            printf("NO\n");
        }
    }
    return 0;
}
```

YES

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
1 1 4 1 1
4
2 0 0 0
4
0 0 2 0
```

Sample Output 1

```
YES
YES
YES
```

Explanation 1

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 %)

```c
#include<stdio.h>
int main()
{
    int T,n,i,left_sum,right_sum;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        int arr[n];
        for(i=0;i<n;i++)
        {
            scanf("%d",&arr[i]);
        }
        left_sum = 0;
        right_sum = 0;
        for(i=0;i<n;i++)
        {
            right_sum += arr[i];
        }
        for(i=0;i<n;i++)
        {
            right_sum -= arr[i];
            if(left_sum == right_sum)
            {
                printf("YES\n");
                break;
            }
            left_sum += arr[i];
        }
        if(left_sum != right_sum)
        {
            printf("NO\n");
        }
    }
    return 0;
}
```

	Input	Expected	Got	
✓	3 5 1 1 4 1 1 4 2 0 0 0 4 0 0 2 0	YES YES YES	YES YES YES	✓
✓	2 3 1 2 3 4 1 2 3 3	NO YES	NO YES	✓

Passed all tests! ✓

Finish review