You are given two integers n and k. Find the greatest integer x, such that, x^k <= n.

Input Format

First line contains number of test cases, T. Next T lines contains integers, n and k.

Constraints

```
1<=T<=10
1<=N<=10^15
1<=K<=10^4
```

Output Format

Output the integer x

Sample Input

```
2
10000 1
1000000000000000 10
```

Sample Output

```
10000
31
```

Explanation

For the first test case, for x=10000, 10000^1=10000=n

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for(int i=0; i<t; i++){
            long n = sc.nextLong();
            long k = sc.nextInt();

            long low = 0;
            long high = n;
            long ans = 0;
            while(low<=high){
                long mid = low+(high-low)/2;
                if(Math.pow(mid,k)==n){
                    ans=mid;
                    break;
                }
                else if(Math.pow(mid,k)<n){
                    ans=mid;
                    low=mid+1;
                }
                else{
                    high=mid-1;
                }
            }
            System.out.println(ans);
        }
    }
}
```

You are given an integer *n*. Find the square root of the number.

Input Format

An integer T, denoting the number of test cases. Each test case contains an integer N.

Constraints

```
1<=T<=10^5 1<=N<=10^12
```

Output Format

Output a floating point number which is square root of the integer N.
(**Note:** Print the answer upto 4 decimal places only).

Sample Input

```
2
100
1000
```

Sample Output

```
10.0000
31.6227
```

Explanation

In first test case, (10.0000)*(10.0000)=100*=n* and hence
10.0000 is the square root of *n*.

```java
import java.util.*;
public class Main {
    public static void main(String
args[]) {
        Scanner sc=new
Scanner(System.in);
        int t=sc.nextInt();
        for(int i=0;i<t;i++){
            int b=sc.nextInt();
            double fn=Math.sqrt(b);
            String
st=String.format("%.5f",fn);
            for(int
j=0;j<st.length();j++){
                if(j!=st.length()-1){

System.out.print(st.charAt(j));
                    }
            }System.out.println();
        }
    }
}
```

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input a number M. Write a function which returns the index on which M is found in the array, in case M is not found -1 is returned. Print the value returned.You can assume that the array is sorted, but you've to optimize the finding process. For an array of size 1024, you can make 10 comparisons at maximum.

1.It reads a number N.
2.Take Another N numbers as input in Ascending Order and store them in an Array.
3.Take Another number M as input and find that number in Array.
4.If the number M is found, index of M is returned else -1 is returned.Print the number returned.

Input Format

Constraints

N cannot be Negative. Range of Numbers N and M can be between -1000000000 to 1000000000

Output Format

Sample Input

```
5
3
5
6
9
78
6
```

Sample Output

```
2
```

Explanation

Array = {3, 5, 6, 9, 78}, target number = 6 . Index of number 6 in the given array = 2. Write Binary search to find the number in given array as discuss in the class.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int[n];
        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        int m = sc.nextInt();

        int low=0;
        int high =n;
        int ans = -1;
        while(low<=high){
            int mid = low+(high-low)/2;
            if(arr[mid]>m){
                high = mid-1;
            }
            else if(arr[mid]<m){
                low=mid+1;
            }
            else{
                ans=mid;
                break;
            }
        }
        System.out.println(ans);
    }
}
```

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input "target", a number. Write a function which prints all pairs of numbers which sum to target.

Input Format

The first line contains input N. Next N lines contains the elements of array and (N+1)th line contains target number.

Constraints

Length of the arrays should be between 1 and 1000.

Output Format

Print all the pairs of numbers which sum to target. **Print each pair in increasing order**.

Sample Input

```
5
1
3
4
2
5
5
```

Sample Output

```
1 and 4
2 and 3
```

Explanation

Find any pair of elements in the array which has sum equal to target element and print them.

```java
import java.util.*;
public class Main {
    public static void main(String
args[]) {
        Scanner sc = new
Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        int target = sc.nextInt();
        for(int i=0; i<n; i++){
            for(int j=i+1; j<n; j++){

if(arr[i]+arr[j]==target){

System.out.println(Math.min(arr[i],ar
r[j])+" and
"+Math.max(arr[i],arr[j]));
                    }
            }
        }
    }
}
```

Take an input N, the size of array. Take N more inputs and store that in an array. Write a function which returns the maximum value in the array. Print the value returned.

1.It reads a number N.

2.Take Another N numbers as input and store them in an Array.

3.calculate the max value in the array and return that value.

Input Format

First line contains integer n as size of array. Next n lines contains a single integer as element of array.

Constraints

N cannot be Negative. Range of Numbers can be between -1000000000 to 1000000000

Output Format

Print the required output.

Sample Input

```
4
2
8
6
4
```

Sample Output

```
8
```

Explanation

Arrays= {2, 8, 6, 4} => Max value = 8 .

```java
import java.util.*;
public class Main {
    public static void main(String
args[]) {
        Scanner sc = new
Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        int max = Integer.MIN_VALUE;
        for(int i=0; i<n; i++){
            arr[i]= sc.nextInt();
            max =
Math.max(arr[i],max);
        }
        System.out.println(max);
    }
}
```

Deepak has a limited amount of money that he can spend on his girlfriend. So he decides to buy two roses for her. Since roses are of varying sizes, their prices are different. Deepak wishes to completely spend that fixed amount of money on buying roses for her.
As he wishes to spend all the money, he should choose a pair of roses whose prices when summed up are equal to the money that he has.
Help Deepak choose such a pair of roses for his girlfriend.

NOTE: If there are multiple solutions print the solution that minimizes the difference between the prices i and j.
After each test case, you must print a blank line.

Input Format

The first line indicates the number of test cases T.
Then, in the next line, the number of available roses, N is given.
The next line will have N integers, representing the price of each rose, a rose that costs less than 1000001.
Then there is another line with an integer M, representing how much money Deepak has.
There is a blank line after each test case.

Constraints

$1 \leq T \leq 100$
$2 \leq N \leq 10000$
Price[i]<1000001

Output Format

For each test case, you must print the message: 'Deepak should buy roses whose prices are i and j.', where i and j are the prices of the roses whose sum is equal do M and $i \leq j$. You can consider that it is always possible to find a solution. If there are multiple solutions print the solution that minimizes the difference between the prices i and j.

Sample Input

```
2
2
40 40
80

5
10 2 6 8 4
10
```

Sample Output

```
Deepak should buy roses whose prices are 40 and 40.
Deepak should buy roses whose prices are 4 and 6.
```

Explanation

Find two such kinds of price of roses which has sum up to equal to Deepak's Money.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        byte t = sc.nextByte();
        for(int i =0; i<t; i++){
            short rose_available = sc.nextShort();
            int [] price = new int[rose_available];
            for(int k=0; k < rose_available; k++){
                price[k]=sc.nextInt();
            }
            int Money = sc.nextInt();
            int [] ans  =new int[2];
            int diff = Integer.MAX_VALUE;
            for(int l = 0; l<rose_available; l++){
                for(int m = l+1; m<rose_available;
m++){
                    int max =
Math.max(price[l],price[m]);
                    int min =
Math.min(price[l],price[m]);
                    if(max+min==Money && diff>max-
min){
                        ans[0]=min;
                        ans[1]=max;
                        diff=ans[1]-ans[0];
                    }
                }
            }
            System.out.println("Deepak should buy
roses whose prices are "+ans[0]+" and
"+ans[1]+".");
        }
    }
}
```

You are provided n numbers (both +ve and -ve). Numbers are arranged in a circular form. You need to find the maximum sum of consecutive numbers.

Input Format

First line contains integer t which is number of test case.
For each test case, it contains an integer n which is the size of array and next line contains n space separated integers denoting the elements of the array.

Constraints

$1<=t<=100$
$1<=n<=1000$
$|A_i| <= 10000$

Output Format

Print the maximum circular sum for each testcase in a new line.

Sample Input

```
1
7
8 -8 9 -9 10 -11 12
```

Sample Output

```
22
```

Explanation

Maximum Circular Sum = 22 (12 + 8 - 8 + 9 - 9 + 10)

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        while (t-- > 0) {
            int n = sc.nextInt();
            int[] arr = new int[n];
            for (int i = 0; i < arr.length;
i++) {
                arr[i] = sc.nextInt();
            }
            System.out.println(maximumsum(arr
));

        }
    }

    public static int maximumsum(int[] arr) {
        int linear_kadane = Kadane(arr);

        int sum = 0;
        for (int i = 0; i < arr.length; i++)
{
            sum = sum + arr[i];
            arr[i] = arr[i] * -1;

        }

        int Inverted_kadane = Kadane(arr);//
12
```

```
        int Cirl_kadane = Inverted_kadane +
sum;
        return Math.max(linear_kadane,
Cirl_kadane);

    }

    public static int Kadane(int[] arr) {
        int ans = Integer.MIN_VALUE;
        int prevsum = 0;
        for (int i = 0; i < arr.length; i++)
{
            prevsum += arr[i];
            ans = Math.max(ans, prevsum);

            if (prevsum < 0) {
                prevsum = 0;
            }
        }
        return ans;

    }

}
```

It is Alex's birthday and she wants to go shopping. She only has 'A' units of money and she wants to spend all of her money. However, she can only purchase one kind of item. She goes to a shop which has 'n' types items with prices $A_0,A_1,A_2,...,A_{n-1}$. The shopkeeper claims that he has atleast 'k' items she can choose from. Help her find out if the shopkeeper is correct or not.

Input Format

The first line contains an integer 'n' denoting the number of items in the shop. The second line contains 'n' space-separated integers describing the respective price of each item. The third line contains an integer 'q' denoting the number of queries. Each of the subsequent lines contains two space-separated integers 'A' and 'k'

Constraints

$1 <= n, A_i, A <= 10^5$ where $0<=i<n$

$1 <= q <= 2*n$
$1 <= k <= n$
The array may contain duplicate elements.

Output Format

For each query, print Yes on a new line if the shopkeeper is correct; otherwise, print No instead.

Sample Input

```
4
100 200 400 100
5
100 2
200 3
500 4
600 4
800 4
```

Sample Output

```
Yes
Yes
No
No
Yes
```

Explanation

In query 1, Alex has 100 units of money. The shopkeeper claims that she can choose to buy from 2 kinds of items i.e. item 1 and item 4 each priced at 100.

In query 2, The shopkeeper claims that she can choose to buy from 3 kinds of items ie item 1 and item 4 each priced at 100(she can buy 1 from either of the two), or item 2 priced at 200(she can buy one)

In query 3, she has 500 units of money. She can either buy item 1 or item 4 (5 of each kind respectively). Thus, she has only 2 kinds of items to choose from.

In query 5, she has 800 units of money. She can either buy item 1 or item 4 (8 of each kind respectively) or item 2(she can buy 4 of these) or item 3(2 of these). Thus, she has 4 kinds of items to choose from

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] item_price = new int [n];
        for(int i=0; i<n; i++){
            item_price[i]=sc.nextInt();
        }
        int query = sc.nextInt();
        int [] money = new int[query];
        int [] option = new int[query];
        for(int j=0; j<query; j++){
            money[j]=sc.nextInt();
            option[j]=sc.nextInt();
            int count=0;
            for(int k=0; k<n; k++){
                if(money[j]%item_price[k]==0){
                    count++;
                }
            }
            if(count>=option[j]){
                System.out.println("Yes");
            }
            else{
                System.out.println("No");
            }
        }
    }
}
```

Given an array nums of length n. We define a running sum of an array as runningSum[i] = sum(nums[0]...nums[i]).

Return the running sum of nums.

Input Format

First line contains an integer n representing number of elements. Next line contains n integers denoting array elements.

Constraints

```
1 <= nums.length <= 1000
-10^6 <= nums[i] <= 10^6
```

Output Format

An integer representing running sum array of the given array

Sample Input

```
4
1 2 3 4
```

Sample Output

```
1 3 6 10
```

Explanation

Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

```java
import java.util.*;
public class Main {
    public static void main (String
args[]) {
        Scanner sc = new
Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        int [] sum = new int [n];
        int sum1=0;
        for(int i=0; i<n; i++){
            arr[i] = sc.nextInt();
            sum1+=arr[i];
            sum[i]=sum1;
            System.out.print(sum[i]+"
");
        }
    }
}
```

Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

Input Format

First line of input contains an integer n representing the length of array n. Next line contains n array elements.

Constraints

1 <= nums.length <= 10^4
-10^4 <= nums[i] <= 10^4
nums is sorted in non-decreasing order.

Output Format

A sorted array representing squares of elements of nums array.

Sample Input

```
5
-4 -1 0 3 10
```

Sample Output

```
0 1 9 16 100
```

Explanation

After squaring, the array becomes [16,1,0,9,100]. After sorting, it becomes [0,1,9,16,100]

```java
import java.util.*;
public class Main {
    public static void main (String
args[]) {
        Scanner sc = new
Scanner(System.in);
        int n = sc.nextInt();
        int [] square = new int[n];
        for(int i=0; i<n; i++){
            square[i]=sc.nextInt();
            square[i]*=square[i];
        }
        for(int i =0; i<n; i++){
            for(int j=i+1; j<n; j++){
                if(square[i]>square[j]){
                    int temp = square[i];
                    square[i]=square[j];
                    square[j]=temp;
                }
            }
        }
        for (int i=0; i<n; i++){
            System.out.print(square[i]+"
");
        }
    }
}
```

Take as input N, a number. Take N more inputs and store that in an array. Write a recursive function which inverses the array. Print the values of inverted array

Input Format

Enter a number N and take N more inputs

Constraints

None

Output Format

Display the values of the inverted array in a space separated manner

Sample Input

```
5
0 2 4 1 3
```

Sample Output

```
0 3 1 4 2
```

Explanation

Swap element with index

for eg : element 4 at index 2 becomes element 2 at index 4

```java
import java.util.*;
public class Main {
    public static void main(String
args[]) {
        // Your Code Here
        Scanner sc = new
Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int[n];
        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        int [] narr = new int[n];
        for(int i=0; i<n; i++){
            int temp=arr[i];
            narr[temp]=i;
        }
        for(int i = 0; i<n; i++){
            System.out.print(narr[i]+"
");
        }
    }
}
```

Raj is a very smart kid who recently started learning computer programming. His coach gave him a cyclic array A having N numbers, and he has to perform Q operations on this array. In each operation the coach would provide him with a number X. After each operation, every element of the cyclic array would be replaced by the sum of itself and the element lying X positions behind it in the cyclic array. All these replacements take place simultaneously. For example, if the cyclic array was [a, b, c, d], then after the operation with X = 1, the new array would be [a+d, b+a, c+b, d+c]. He needs to output the sum of the elements of the final array modulus 10^9+7. He made a program for it but it's not very efficient. You know he is a beginner, so he wants you to make an efficient program for this task because he doesn't want to disappoint his coach.

Input Format

The first line of each test file contains a integer N. The next line contains N space separated integers which represent the elements of the cyclic array. The third line contains a integer Q representing the number of operations that will be applied to the array. Finally, Q lines follow, each one containing an integer X .

Constraints

```
1 <= N <= 100000
1 <= Ai<= 10^9
0 <= Q <= 1000000
0 <= X < N
```

Output Format

Your program should output to the standard output stream the sum of the elements of the final array modulus 10^9+7.

Sample Input

```
5
1 2 3 4 5
2
1
0
```

Sample Output

```
60
```

Explanation

```
After the 1st operation (X = 1), the array would be [1+5, 2+1,
3+2, 4+3, 5+4] =[6, 3, 5, 7, 9]
After 2nd operation (X = 0), the array would be [6+6, 3+3, 5+5,
7+7, 9+9] =[12, 6, 10, 14, 18]
Thus the correct answer would equal to = (12+6+10+14+18) %
(10^9+7) = 60
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] num = new int [n];
        for(int i=0; i<n; i++){
            num[i]=sc.nextInt();
        }
        int operation = sc.nextInt();
        for(int j=0; j<operation; j++){
            int x = sc.nextInt();
            int [] ans = new int [n];
            for(int i=0; i<n; i++){
                if(i-x>=0){
                    ans[i]=num[i]+num[i-x];
                }
                else{
                    ans[i]=num[i]+num[i+n-x];
                }
            }
            for(int k=0; k<n; k++){
                num[k]=ans[k];
            }
        }
        int sum=0;
        for(int i=0; i<n; i++){
            sum+=num[i];
        }
        System.out.print((int)(sum %
(Math.pow(10,9)+7)));
    }
}
```

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input "target", a number. Write a function which prints all triplets of numbers which sum to target.

Input Format

First line contains input N.
Next line contains N space separated integers denoting the elements of the array.
The third line contains a single integer T denoting the target element.

Constraints

Length of Array should be between 1 and 1000.

Output Format

Print all the triplet present in the array in a new line each. The triplets must be printed as A, B and C where A,B and C are the elements of the triplet (A<=B<=C) and all triplets must be printed in sorted order. Print only unique triplets.

Sample Input

```
9
5 7 9 1 2 4 6 8 3
10
```

Sample Output

```
1, 2 and 7
1, 3 and 6
1, 4 and 5
2, 3 and 5
```

Explanation

Array = {5, 7, 9, 1, 2, 4, 6 ,8 ,3}. Target number = 10. Find any three number in the given array which sum to target number.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        Arrays.sort(arr);
        int target = sc.nextInt();
        for(int i=0; i<n; i++){
            for(int j=i+1; j<n; j++){
                for(int k=j+1; k<n; k++){
                    int [] tar = new int [3];

if(arr[i]+arr[j]+arr[k]==target){
                        tar[0]=arr[i];
                        tar[1]=arr[k];
                        tar[2]=arr[j];
                        Arrays.sort(tar);

System.out.println(tar[0]+", "+tar[1]+" and
"+tar[2]);
                    }
                }
            }
        }

    }
}
```

Take as input N, a number. Take N more inputs to form an array. The array contains only 0 and 1. Sort the array in a single scan.

Input Format

Enter the size of the array N and input N more numbers and store in the array

Constraints

Output Format

Display the sorted array

Sample Input

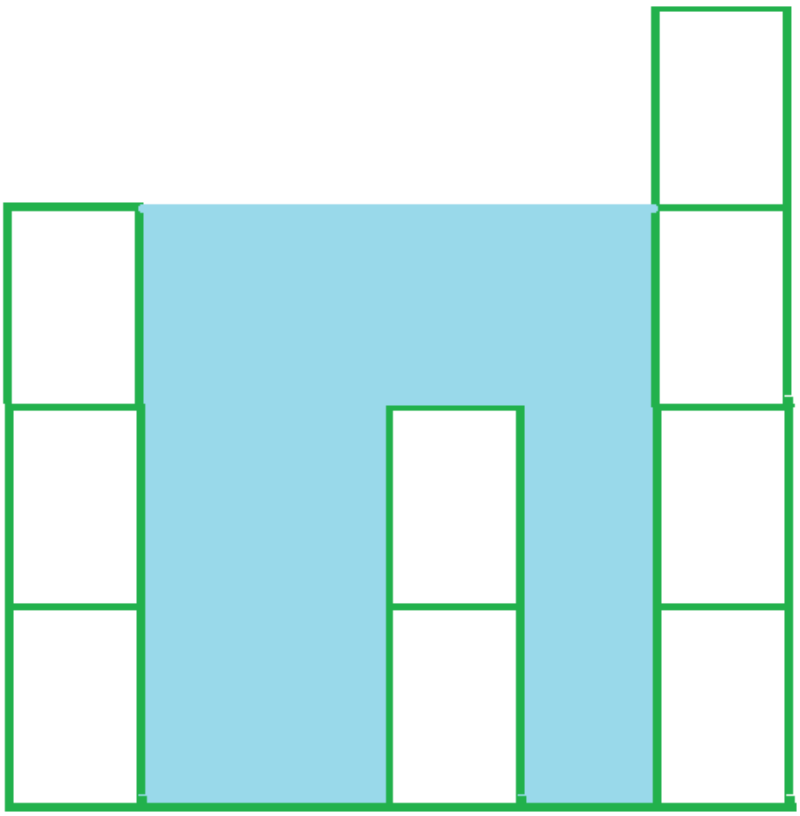
```
5
1
1
1
0
0
```

Sample Output

```
0 0 1 1 1
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int len = sc.nextInt();
        int [] arr = new int[len];
        for(int i =0;i<len;i++){
            arr[i]=sc.nextInt();
        }
        for(int i = 0;i<len;i++){
            for(int j = i+1;j<len;j++){
                if(arr[i]>arr[j]){
                    int temp=arr[i];
                    arr[i]=arr[j];
                    arr[j]=temp;
                }
            }
        }
    for(int i =0; i<len;i++){
        System.out.print(arr[i]+" ");
    }
    }
}
```

You are given an input array whose each element represents the height of a line towers. The width of every tower is 1. It starts raining. Water is filled between the gap of towers if possible. You need to find how much water filled between these given towers.



Example : **Bars for input {3, 0, 0, 2, 0, 4}**
Total trapped water = 3 + 3 + 1 + 3 = 10

Input Format

The first line consists of number of test cases T. Each test case consists an integer N as number of towers and next line contains the N space separated integers.

Constraints

1 <= N <= 1000000 1 <= t <= 10 0 <= A[i] <= 10000000

Output Format

Print how much unit of water collected among towers for each test case.

Sample Input

```
2
6
3  0  0  2  0  4
12
0  1  0  2  1  0  1  3  2  1  2  1
```

Sample Output

```
10
6
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t=sc.nextInt();
        for(int i=0;i<t;i++){
            int n = sc.nextInt();
            int [] arr= new int[n];
            for(int j=0; j<n; j++){
                arr[j]=sc.nextInt();
            }
            int [] lmax = new int [n];
            int [] rmax = new int [n];
            lmax[0]=arr[0];
            rmax[n-1]=arr[n-1];
            for(int k =1;k<n;k++){
                lmax[k]=Math.max(lmax[k-
1],arr[k]);
                rmax[n-k-1]=Math.max(rmax[n-
k],arr[n-k-1]);
            }
            int ans =0;
            for(int l = 0; l<n;l++){

ans+=Math.min(lmax[l],rmax[l])-arr[l];
            }
            System.out.println(ans);
        }
    }
}
```

You are provided two sorted arrays. You need to find the maximum length of bitonic subsequence. You need to find the sum of the maximum sum path to reach from beginning of any array to end of any of the two arrays. You can switch from one array to another array only at common elements.

Input Format

First line contains integer t which is number of test case. For each test case, it contains two integers n and m which is the size of arrays and next two lines contains n and m space separated integers respectively.

Constraints

1<=t<=100 1<=n,m<=100000

Output Format

Print the maximum path.

Sample Input

```
1
8 8
2 3 7 10 12 15 30 34
1 5 7 8 10 15 16 19
```

Sample Output

```
122
```

Explanation

122 is sum of 1, 5, 7, 8, 10, 12, 15, 30, 34

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        while (t-- > 0) {

            int n = sc.nextInt();
            int m = sc.nextInt();
            int[] arr = new int[n];
            int[] brr = new int[m];
            for (int i = 0; i < arr.length; i++) {

                arr[i] = sc.nextInt();
            }
            for (int i = 0; i < brr.length; i++) {
                brr[i] = sc.nextInt();
            }
            System.out.println(pathSum(arr, brr));
        }

    }

    public static int pathSum(int[] arr, int[] brr)
{
        int i = 0;// arr
        int j = 0;// brr
        int p = 0;// arr start
        int q = 0;// brr start
        int ans = 0;
        while (i < arr.length && j < brr.length) {
            if (arr[i] < brr[j]) {
                i++;
            } else if (arr[i] > brr[j]) {
                j++;
            } else {
                int sum1 = 0;
```

```
                int sum2 = 0;
                for (int k = p; k <= i; k++) {
                    sum1 = sum1 + arr[k];

                }
                for (int k = q; k <= j; k++) {
                    sum2 += brr[k];

                }
                ans = ans + Math.max(sum1, sum2);
                i++;
                j++;
                p = i;
                q = j;
            }

        }

        if (i == arr.length) {
            for (int k = q; k < brr.length; k++) {
                ans = ans + brr[k];
            }
        }

        if (j == brr.length) {
            for (int k = p; k < arr.length; k++) {
                ans = ans + arr[k];
            }
        }
        return ans;

    }

}
```

Given an array Arr[], Treat each element of the array as the digit and whole array as the number. Implement the next permutation, which rearranges numbers into the numerically next greater permutation of numbers.

If such arrangement is not possible, it must be rearranged as the lowest possible order ie, sorted in an ascending order.

Note: The replacement must be in-place, do not allocate extra memory.

Input Format

The First Line contains the Number of test cases T.
Next Line contains an Integer N, number of digits of the number.
Next Line contains N-space separated integers which are elements of the array 'Arr'.

Constraints

$1 <= T <= 100$
$1 <= N <= 1000$
$0 <= A_i <= 9$

Output Format

Print the Next Permutation for each number separated by a new Line.

Sample Input

```
2
3
1 2 3
3
3 2 1
```

Sample Output

```
1 3 2
1 2 3
```

Explanation

Possible permutations for {1,2,3} are {1,2,3} , {1,3,2} , {2,1,3} , {2,3,1}, {3,1,2} and {3,2,1}. {1,3,2} is the immediate next permutation after {1,2,3}.
For the second testcase , {3,2,1} is the last configuration so we print the first permutation as its next permutation.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for(int i= 0; i<t; i++){
            int n = sc.nextInt();
            int [] digit = new int [n];
            for (int j =0; j<n; j++){
                digit[j]=sc.nextInt();
            }
            if(n<=1){
                System.out.println(digit[0]);
            }
            else{
                int left_high = n-2;
                while(left_high >=0 &&
digit[left_high]>=digit[left_high+1]){
                    left_high--;
                }
                if(left_high>=0){
                    int r_g_l = n-1;
                    while(r_g_l >=0 &&
digit[r_g_l]<=digit[left_high]){
                        r_g_l--;
                    }
                    swap(digit,left_high,r_g_l);
                }
                reverse(digit,left_high+1,n-1);
                display(digit,n);
                System.out.println();
            }
        }
    }
    public static void swap(int []digit, int
left_high, int r_g_l){
        int temp = digit[left_high];
```

```
        digit[left_high]=digit[r_g_l];
        digit[r_g_l]=temp;
    }
    public static void reverse(int [] digit, int i,
int j){
        while(i<j){
            swap(digit,i++,j--);
        }
    }

    public static void display(int [] digit, int
n){
        for(int i = 0; i<n; i++){
            System.out.print(digit[i]+" ");
        }
    }
}
```

Ramu often uses public transport. The transport in the city is of two types: cabs and rickshaws. The city has n rickshaws and m cabs, the rickshaws are numbered by integers from 1 to n, the cabs are numbered by integers from 1 to m.

Public transport is not free. There are 4 types of tickets:

A ticket for one ride on some rickshaw or cab. It costs c1 ruppees;
A ticket for an unlimited number of rides on some rickshaw or on some cab. It costs c2 ruppees;
A ticket for an unlimited number of rides on all rickshaws or all cabs. It costs c3 ruppees;
A ticket for an unlimited number of rides on all rickshaws and cabs. It costs c4 ruppees.

Ramu knows for sure the number of rides he is going to make and the transport he is going to use. He asked you for help to find the minimum sum of ruppees he will have to spend on the tickets.

Input Format

Each Test case has 4 lines which are as follows:

The first line contains four integers c1, c2, c3, c4 ($1 \le c1, c2, c3, c4 \le 1000$) — the costs of the tickets.
The second line contains two integers n and m ($1 \le n, m \le 1000$) — the number of rickshaws and cabs Ramu is going to use.
The third line contains n integers ai ($0 \le ai \le 1000$) — the number of times Ramu is going to use the rickshaw number i.
The fourth line contains m integers bi ($0 \le bi \le 1000$) — the number of times Ramu is going to use the cab number i.

Constraints

1 <= T <= 1000 , where T is no of testcases
$1 \le c1, c2, c3, c4 \le 1000$
$1 \le n, m \le 1000$
$0 \le a_i, b_i \le 1000$

Output Format

For each testcase , print a single number - the minimum sum of rupees Ramu will have to spend on the tickets in a new line.

Sample Input

```
2
1 3 7 19
2 3
2 5
4 4 4
4 3 2 1
1 3
798
1 2 3
```

Sample Output

```
12
1
```

Explanation

For the first testcase ,
The total cost of rickshaws = min(min(2 * 1, 3) + min(5 * 1, 3), 7)
= min(5, 7) = 5
The total cost of cabs = min(min(4 * 1, 3) + min(4 * 1, 3) + min(4 * 1, 3) , 7) = min (9, 7) = 7
Total final cost = min(totalCabCost + totalRickshawCost , c4)

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for(int j=0; j<t; j++){
            int c1= sc.nextInt();
            int c2= sc.nextInt();
            int c3= sc.nextInt();
            int c4= sc.nextInt();

            int rick = sc.nextInt();
            int cab= sc.nextInt();
            int [] rickshaw = new int [rick];
            int [] cabs = new int [cab];
            int r_cost = 0;
            int c_cost = 0;
            int Total_cost = 0;
            for(int i=0; i<rick; i++){
                rickshaw[i]=sc.nextInt();
                r_cost +=
Math.min(rickshaw[i]*c1,c2);
            }
            r_cost=Math.min(r_cost,c3);
            for(int i=0; i<cab; i++){
                cabs[i]=sc.nextInt();
                c_cost+= Math.min(cabs[i]*c1,c2);
            }
            c_cost=Math.min(c_cost,c3);
            Total_cost =
Math.min(c_cost+r_cost,c4);
            System.out.println(Total_cost);
        }
    }
}
```

You will be given an array containing only 0s, 1s and 2s. you have sort the array in linear time that is O(N) where N is the size of the array.

Input Format

The first line contains `N`, which is the size of the array. The following N lines contain either `0`, or `1`, or `2`.

Constraints

1 <= N <= 10^6
Each input element x, such that x ∈ { 0, 1, 2 }.

Output Format

Output the sorted array with each element separated by a newline.

Sample Input

```
5
0
1
2
1
2
```

Sample Output

```
0
1
1
2
2
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int [n];
        int count0 = 0;
        int count1 = 0;
        int count2 = 0;
        for(int i = 0; i<n; i++){
            arr[i]= sc.nextInt();
            if(arr[i]==0){
                count0++;
            }
            else if(arr[i] == 1){
                count1++;
            }
            else{
                count2++;
            }

            }
        for(int i = 0; i<n; i++){
            if(i<count0){
                System.out.println(0);
            }
            else if(i<(count0+count1)){
                System.out.println(1);
            }
            else{
                System.out.println(2);
            }

        }
        }
    }
```

Given an array arr of n integers where n > 1, return an array output such that output[i] is equal to the product of all the elements of arr except arr[i].

Input Format

First line contains integer **N** as size of array.
Next line contains a **N** integer as element of array.

Constraints

arr[i]<=10000000

Output Format

print output array

Sample Input

```
4
1 2 3 4
```

Sample Output

```
24 12 8 6
```

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        long [] arr = new long[n];
        long [] prod = new long[n];
        for(int i =0; i<n; i++){
           arr[i]=sc.nextInt();
        }
        for(int i =0; i<n; i++){
            long prod1=1;
           for(int j =0; j<n; j++){
               if(j!=i){
                   prod1*=arr[j];
               }
}

           prod[i]=prod1;
           System.out.print(prod1+" ");
        }
        }
}
```

Take as input N, the size of an array. Take N more inputs and store that in an array. Take another number's input as M. Write a function which returns the index on which M is found in an array, in case M is not found -1 is returned. Print the value returned.

1. It reads a number N.
   2.Take Another N numbers as an input and store them in an Array.
2. Take another number M as an input.
3. If M is found in the Array the index of M is returned else -1 is returned and print the value returned.

Input Format

Constraints

N cannot be Negative. Range of Numbers can be between -1000000000 to 1000000000. M can be between -1000000000 to 1000000000.

Output Format

Sample Input

```
5
2
4
6
9
17
17
```

Sample Output

```
4
```

Explanation

Given array = {2, 4, 6, 9, 17}. Target number = 17. Index = 4.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        int target = sc.nextInt();
        int index=-1;
        for(int i=0; i<n; i++){
            if(arr[i]==target){
                index=i;
            }
        }
        System.out.print(index);
    }
}
```

Take as input N, the size of array. Take N more inputs and store that in an array. Write a function that reverses the array. Print the values in reversed array.

1.It reads a number N.
2.Take Another N numbers as input and store them in an Array.
3.Reverse the elements in the Array.
4.Print the reversed Array.

Input Format

First-line contains a single integer n denoting the size of the array.
Next, N line contains a single integer denoting the elements of the array.

Constraints

N cannot be Negative. Range of Numbers can be between -1000000000 to 1000000000.

Output Format

Print the elements of the reversed array

Sample Input

```
5
0
4
6
8
9
```

Sample Output

```
9
8
6
4
0
```

Explanation

In the sample case , arr=[0,4,6,8,9] is reversed to arr= [9,8,6,4,0].

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int [] arr = new int [n];
        for(int i=0; i<n; i++){
            arr[i]=sc.nextInt();
        }
        int i=0;
        int j=n-1;
        while(i<j){
            int temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
            i++;
            j--;
        }
        for(i=0; i<n; i++){
            System.out.println(arr[i]);
        }
    }
}
```