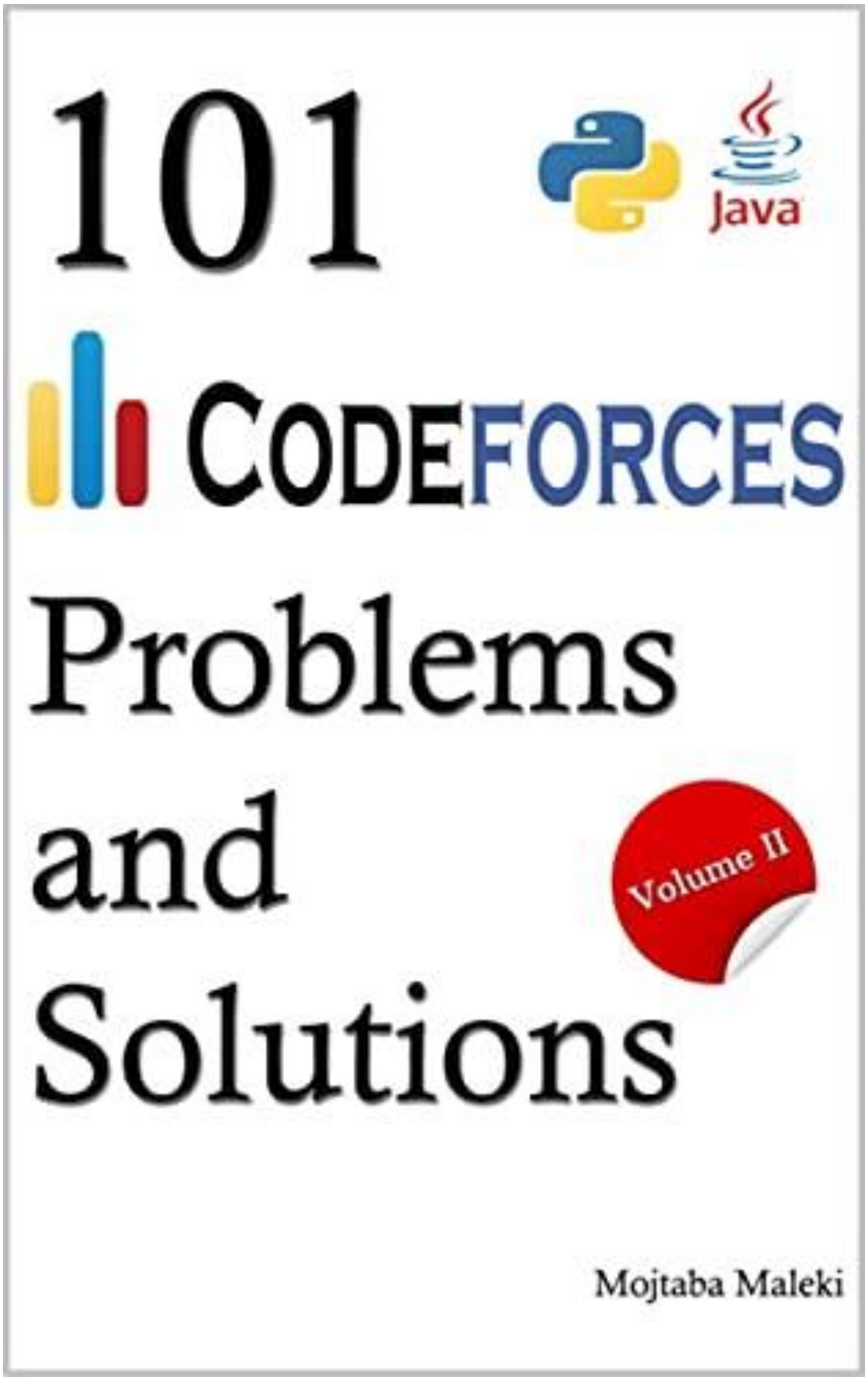


101 Codeforces Problems and Solutions

Volume II

Mojtaba Maleki

101 Codeforces Problems and solutions 2

Mojtaba Maleki

ISBN:

DEDICATION

I dedicate this book to my hardworking family who never give up,
to whom taught me never give up.

CONTENTS

INTRODUCTION

This book is about programming problems and solutions that contain 101 problems from one of the best programming test websites Codeforces "**https://codeforces.com**". All of the problems have been solved with simple solutions.

Most of the solutions are in java, but there are some solutions had been solved by python.

The best way of using this book is at first reading the problems, after a full understanding of problems go to the brainstorming part and write down your idea. In the next step, solve the problem. If you were successful in solving that, it is better to see our solutions too. If not, the best way is to check our answers and try again and again until you reach the answer.

The book is sorted from simple problems to hard ones and contains the problems in six 800, 900, 1000, 1100, 1200, over 1300 score ranges.

This book is a great source for people or students who are at the beginning of programming and want to improve their programming skills.

To my dear readers,

Think big and out of the box.

Wish you all the best.

Thank you for choosing me

Mojtaba Maleki

1. 800 SCORES PROBLEMS

Ridbit starts with an integer *n*. In one move, he can perform one of the following operations:

- divide *n* by one of its proper divisors, or subtract 1 from *n* if *n* is greater than 1.

A proper divisor is a divisor of a number, excluding itself. For example, 1, 2, 4, 5, and 10 are proper divisors of 20, but 20itself is not.

What is the minimum number of moves Ridbit is required to make to reduce *n* to 1?

Input

The first line contains a single integer $t (1\leq t \leq 1000)$ — the number of test cases. The only line of each test case contains a single integer $n (1\leq n\leq 10$^9).

Output

For each test case, output the minimum number of moves required to reduce *n* to 1.

Answer (java):

```java
package com.company;

import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        for (int i = 0; i < number; i++) {
            int numbers = input.nextInt();
            if (numbers==3){
                System.out.println("2");
            }else if (numbers==2){
                System.out.println("1");
            }else if (numbers==1){
                System.out.println("0");
            }else if (numbers%2==1){
                System.out.println("3");
            }else {
                System.out.println("2");
            }
        }
    }
}
```

Given an array a of length n, tell us whether it has a non-empty subsequence such that the product of its elements is not a perfect square. A sequence b is a subsequence of an array a if b can be obtained from a by deleting some (possibly zero) elements.

Input

The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n($1 \le n \le 100$) — the length of the array a. The second line of each test case contains n integers a_1, a_2, …, a_n ($1 \le a_i \le 10^4$) — the elements of the array a.

Output

If there's a subsequence of a whose product isn't a perfect square, print "YES". Otherwise, print "NO".

Answer (java):

```java
package com.company;
import java.util.*;
import java.io.*;

public class Main {

    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int sum=0;

        for (int i=0;i<n;i++){
            int number= input.nextInt();
            sum=0;
            for (int j=0;j<number;j++){
                int numbers= input.nextInt();
                //System.out.println("numbers:
"+numbers);
                if
(Math.round(Math.sqrt(numbers))!=Math.sqrt(numbers)){
                    sum=sum+1;
                }
            }
            if (sum==0){
                System.out.println("NO");
            }else {
                System.out.println("YES");
            }
        }
    }
}
```

You are given a string s, consisting of n letters, each letter is either 'a' or 'b'. The letters in the string are numbered from 1 to n. $s[l;r]$ is a continuous substring of letters from index l to r of the string inclusive. A string is called balanced if the number of letters 'a' in it is equal to the number of letters 'b'. For example, strings "baba" and "aabbab" are balanced and strings "aaab" and "b" are not.

Find any non-empty balanced substring $s[l;r]$ of string s. Print its l and r ($1 \le l \le r \le n$). If there is no such substring, then print -1 -1.

Input

The first line contains a single integer $t (1 \le t \le 1000)$ — the number of testcases. Then the descriptions of t testcases follow.

The first line of the testcase contains a single integer n ($1 \le n \le 50$) — the length of the string.

The second line of the testcase contains a string s, consisting of n letters, each letter is either 'a' or 'b'.

Output

For each testcase print two integers. If there exists a non-empty balanced substring $s[l;r]$, then print l r ($1 \le l \le r \le n$). Otherwise, print -1 -1.

Answer (java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Vector;
public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        String num = input.nextLine();
        int number = Integer.parseInt(num);
        for (int i=0;i<number;i++){
            String num2= input.nextLine();
            int numbers=Integer.parseInt(num2);
            String word= input.nextLine();
            int left=word.indexOf("ab");
            int right=word.indexOf("ba");
            if (left!=-1) {
                System.out.println((left+1) + " " +
(left + 2));
            }else if (right!=-1){
                System.out.println((right+1)+"
"+(right+2));
            }else {
                System.out.println("-1 -1");
            }
        }
    }
}
```

Polycarp knows that if the sum of the digits of a number is divisible by 3, then the number itself is divisible by 3. He assumes that the numbers, the sum of the digits of which is divisible by 4, are also somewhat interesting. Thus, he considers a positive integer n interesting if its sum of digits is divisible by 4.

Help Polycarp find the nearest larger or equal interesting number for the given number a. That is, find the interesting number n such that $n \ge a$ and n is minimal.

Input

The only line in the input contains an integer a ($1 \le a \le 1000$).

Output

Print the nearest greater or equal interesting number for the given number a. In other words, print the interesting number n such that $n \ge a$ and n is minimal.

Answer (java):

```java
package com.company;
import java.util.*;
import java.io.*;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner sc = new Scanner(System.in);
        int number = sc.nextInt();
        int num=number;
        int nu1=num;
        int digit=0;
        int sum=0;

        for (int i=number;i<100000000;i++){
            int i2=i;
            while(i2 > 0)
            {
         //finds the last digit of the given number
                digit = i2 % 10;
         //adds last digit to the variable sum
                sum = sum + digit;
         //removes the last digit from the number
                i2 = i2 / 10;
            }

            if (sum%4==0){
                System.out.println(i);
                break;
            }
            sum=0;
        }
        //System.out.println(sum);
    }
}
```

There are n students in a university. The number of students is even. The i-th student has programming skill equal to a_i.

The coach wants to form $n/2$ teams. Each team should consist of exactly two students, and each student should belong to exactly one team. Two students can form a team only if their skills are equal (otherwise they cannot understand each other and cannot form a team).

Students can solve problems to increase their skill. One solved problem increases the skill by one.

The coach wants to know the minimum total number of problems students should solve to form exactly $n/2$

teams (i.e. each pair of students should form a team). Your task is to find this number.

Input

The first line of the input contains one integer n ($2 \le n \le 100$) — the number of students. It is guaranteed that n is even.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$), where a_i is the skill of the i-th student.

Output

Print one number — the minimum total number of problems students should solve to form exactly $n/2$ teams.

Answer(java):

```
package com.company;
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        int total=0;
        int []arr=new int[number];
        for (int i=0;i<number;i++){
            arr[i]= input.nextInt();
        }
        Arrays.sort(arr);
        for (int j=0;j<number;j++){
            if (j+1==number){
                continue;
            }
            if (arr[j]!=arr[j+1]){
                total=total+arr[j+1]-arr[j];
            }
            j=j+1;
        }
        System.out.println(total);
    }
}
```

There is an infinite 2-dimensional grid. The robot stands in cell $(0,0)$ and wants to reach cell (x,y). Here is a list of possible commands the robot can execute:

- move north from cell (i,j) to $(i,j+1)$;
- move east from cell (i,j) to $(i+1,j)$;
- move south from cell (i,j) to $(i,j-1)$;
- move west from cell (i,j) to $(i-1,j)$;
- stay in cell (i,j).

The robot wants to reach cell (x,y) in as few commands as possible. However, he can't execute the same command two or more times in a row.

What is the minimum number of commands required to reach (x,y) from $(0,0)$?

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of testcases.

Each of the next t lines contains two integers x and y ($0 \le x,y \le 10$^4) — the destination coordinates of the robot.

Output

For each testcase print a single integer — the minimum number of commands required for the robot to reach (x,y) from $(0,0)$ if no command is allowed to be executed two or more times in a row.

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        for (int i=0;i<number;i++){
            int numbers = input.nextInt();
            int numbers2 = input.nextInt();

            if (numbers==numbers2){
                System.out.println(numbers+numbers2);
            }else{

System.out.println(2*Math.max(numbers,numbers2)-1);
            }
        }
    }
}
```

You can not just take the file and send it. When Polycarp trying to send a file in the social network "Codehorses", he encountered an unexpected problem. If the name of the file contains three or more "x" (lowercase Latin letters "x") in a row, the system considers that the file content does not correspond to the social network topic. In this case, the file is not sent and an error message is displayed.

Determine the minimum number of characters to remove from the file name so after that the name does not contain "xxx" as a substring. Print 0 if the file name does not initially contain a forbidden substring "xxx".

You can delete characters in arbitrary positions (not necessarily consecutive). If you delete a character, then the length of a string is reduced by 1. For example, if you delete the character in the position 2 from the string "exxxii", then the resulting string is "exxii".

Input

The first line contains integer n ($3 \le n \le 100$)— the length of the file name. The second line contains a string of length n consisting of lowercase Latin letters only — the file name.

Output

Print the minimum number of characters to remove from the file name so after that the name does not contain "xxx" as a substring. If initially the file name dost not contain a forbidden substring "xxx", print 0.

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String nu = input.nextLine();
        int number = Integer.parseInt(nu);
        int sum=0;
        int total=0;

        String word1 = input.nextLine();
        String word = word1+'0';
        if (word.contains("xxx")) {

            for (int i=0;i<number+1;i++) {
                if (word.charAt(i)=='x'){
                    sum=sum+1;
                }
                if (word.charAt(i)!='x'){
                    if (sum>=3){
                        total=total+sum-2;
                    }
                    sum=0;
                }
            }
            System.out.println(total);
        }else {
            System.out.println("0");
        }
    }
}
```

There are n cats in a line, labeled from 1 to n, with the i-th cat at position i. They are bored of gyrating in the same spot all day, so they want to reorder themselves such that no cat is in the same place as before. They are also lazy, so they want to minimize the total distance they move. Help them decide what cat should be at each location after the reordering. For example, if there are 3 cats, this is a valid reordering: $[3,1,2]$. No cat is in its original position. The total distance the cats move is $1+1+2=4$ as cat 1 moves one place to the right, cat 2 moves one place to the right, and cat 3 moves two places to the left.

Input

The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases. Then t test cases follow. The first and only line of each test case contains one integer n $(2 \le n \le 100)$ — the number of cats. It can be proven that under the constraints of the problem, an answer always exist.

Output

Output t answers, one for each test case. Each answer consists of n integers — a permutation with the minimum total distance. If there are multiple answers, print any.

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number=input.nextInt();

        for (int i=0;i<number;i++){
            int numbers= input.nextInt();
            int[]arr=new int[numbers];
            int[]arr1=new int[numbers];
            int[]arr2=new int[numbers];
            int[]arr3=new int[numbers];
            for (int j=0;j<numbers;j++){
                arr[j]=j+1;
            }
            arr1=arr;
            arr2=arr;
            arr3=arr2;
            int original=arr1[numbers-1];

            for (int m=0;m<numbers;m++){
                if (m+1==numbers){
                    continue;
                }
                int test=arr1[m];

                arr[m]=arr2[m+1];
                arr[m+1]=test;
                m=m+1;

            }
            if (numbers%2==1){
                arr[numbers-1]=arr[numbers-2];
                arr[numbers-2]=original;

            }
            for (int l=0;l<numbers;l++){
                arr[l]+" ");
                System.out.print(arr[l]+" ");
            }
            System.out.println(" ");

        }
    }
}
```

Today, Mezo is playing a game. Zoma, a character in that game, is initially at position $x=0$. Mezo starts sending n commands to Zoma. There are two possible commands:

- 'L' (Left) sets the position $x:=x-1$;
- 'R' (Right) sets the position $x:=x+1$.

Unfortunately, Mezo's controller malfunctions sometimes. Some commands are sent successfully and some are ignored. If the command is ignored then the position x doesn't change and Mezo simply proceeds to the next command.

For example, if Mezo sends commands "LRLR", then here are some possible outcomes (underlined commands are sent successfully):

- "LRLR" — Zoma moves to the left, to the right, to the left again and to the right for the final time, ending up at position 0 ;
- "LRLR" — Zoma recieves no commands, doesn't move at all and ends up at position 0 as well;
- "LRLR" — Zoma moves to the left, then to the left again and ends up in position -2.

Mezo doesn't know which commands will be sent successfully beforehand. Thus, he wants to know how many different positions may Zoma end up at.

Input

The first line contains n $(1\leq n\leq 10^5)$— the number of commands Mezo sends. The second line contains a string $s$of n commands, each either 'L' (Left) or 'R' (Right).

Output
Print one integer — the number of different positions Zoma may end up at.

Answer(java):

```
package com.company;
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        String nu = input.nextLine();
        int number=Integer.parseInt(nu);
        String navigation=input.nextLine();
        int left=0;
        int right=0;
        for (int i=0;i<number;i++){
            if (navigation.charAt(i)=='L'){
                left=left-1;
            }else {
                right=right+1;
            }
        }
        System.out.println(right-left+1);
    }
}
```

Recently Anton found a box with digits in his room. There are k_2 digits 2, k_3 digits 3, k_5 digits 5 and k_6 digits 6.

Anton's favorite integers are 32 and 256. He decided to compose this integers from digits he has. He wants to make the sum of these integers as large as possible. Help him solve this task!

Each digit can be used no more than once, i.e. the composed integers should contain no more than k_2 digits 2, k_3 digits 3 and so on. Of course, unused digits are not counted in the sum.

Input

The only line of the input contains four integers k_2, k_3, k_5 and k_6 — the number of digits 2, 3, 5 and 6 respectively ($0 \le k_2, k_3, k_5, k_6 \le 5 \cdot 10^6$).

Output

Print one integer — maximum possible sum of Anton's favorite integers that can be composed using digits from the box.

Answer (java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {


    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int k2= input.nextInt();
        int k3= input.nextInt();
        int k5= input.nextInt();
        int k6= input.nextInt();
        int sum1=0;
        int sum2=0;
        int sum=0;

        int []a256=new int[3];
        int []a32=new int[2];
        a256[0]=k2;
        a256[1]=k5;
        a256[2]=k6;

        Arrays.sort(a256);
        sum1=a256[0]*256;
        //System.out.println(sum1);

        a32[0]=k2-a256[0];
        a32[1]=k3;
        Arrays.sort(a32);
        sum2=(a32[0])*32;
        //System.out.println(sum2);
        sum=sum1+sum2;
        System.out.println(sum);


    }
}
```

You are a coach of a group consisting of n students. The i-th student has programming skill a_i. All students have distinct programming skills. You want to divide them into teams in such a way that:

- No two students I and j such that $|a_i - a_j| = 1$ belong to the same team (i.e. skills of each pair of students in the same team have the difference strictly greater than 1);
- the number of teams is the minimum possible.

You have to answer q independent queries.

Input

The first line of the input contains one integer q ($1 \le q \le 100$) — the number of queries. Then q queries follow.

The first line of the query contains one integer n ($1 \le n \le 100$) — the number of students in the query. The second line of the query contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$, all a_i are distinct), where a_i is the programming skill of the I -th student.

Output

For each query, print the answer on it — the minimum number of teams you can form if no two students I and j such that $|a_i - a_j| = 1$ may belong to the same team (i.e. skills of each pair of students in the same team has the difference strictly greater than 1)

Answer (java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String args[]) throws
IOException {

        Scanner input= new Scanner(System.in);
        int number= input.nextInt();
        int sum=0;


        for (int i=0;i<number;i++) {
            int numbers = input.nextInt();

            int[] arr = new int[numbers];
            for (int j = 0; j < numbers; j++) {
                arr[j] = input.nextInt();
            }
            Arrays.sort(arr);

//System.out.println(Arrays.toString(arr));

            for (int k=0;k<numbers;k++){
                if ( k+1==numbers){
                    System.out.println("1");
                }
                if (k+1==numbers){
                    continue;
                }

                if (arr[k+1]-arr[k]==1){
                    System.out.println("2");
                    sum=sum+1;
                    break;
                }
            }
        }
    }
}
```

Dawid has four bags of candies. The i-th of them contains a_i candies. Also, Dawid has two friends. He wants to give each bag to one of his two friends. Is it possible to distribute the bags in such a way that each friend receives the same amount of candies in total?

Note, that you can't keep bags for yourself or throw them away, each bag should be given to one of the friends.

Input

The only line contains four integers a_1, a_2, a_3 and a_4 ($1 \le a_i \le 100$) — the numbers of candies in each bag.

Output

Output YES if it's possible to give the bags to Dawid's friends so that both friends receive the same amount of candies, or NO otherwise. Each character can be printed in any case (either uppercase or lowercase).

Answer (java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int []arr=new int[4];
        arr[0]= input.nextInt();
        arr[1]= input.nextInt();
        arr[2]= input.nextInt();
        arr[3]= input.nextInt();
        int result =arr[0]+arr[1]+arr[2]+arr[3];
        Arrays.sort(arr);
        int last= arr[3];
        if (result%2==0){
            if (arr[0]+arr[1]==result/2 ||
arr[0]+arr[2]==result/2 || arr[0]+arr[3]==result/2 ||
arr[1]+arr[2]==result/2 || arr[1]+arr[3]==result/2 ||
arr[2]+arr[3]==result/2 || last==result/2 ){
                System.out.println("YES");
            }else {
                System.out.println("NO");
            }
        }else {
            System.out.println("NO");
        }
    }
}
```

You are the gym teacher in the school.

There are n students in the row. And there are two rivalling students among them. The first one is in position a, the second in position b. Positions are numbered from 1 to n from left to right.

Since they are rivals, you want to maximize the distance between them. If students are in positions p and s respectively, then distance between them is $|p-s|$.

You can do the following operation at most x times: choose two adjacent (neighbouring) students and swap them.

Calculate the maximum distance between two rivalling students after at most x swaps.

Input

The first line contains one integer t $(1 \le t \le 100)$ — the number of test cases.

The only line of each test case contains four integers n, x, a and b $(2 \le n \le 100,\ 0 \le x \le 100,\ 1 \le a, b \le n,\ a \ne b)$ — the number of students in the row, the number of swaps which you can do, and positions of first and second rivaling students respectively.

Output

For each test case print one integer — the maximum distance between two rivaling students which you can obtain.

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String args[]) throws
IOException {

        Scanner input= new Scanner(System.in);
        int num= input.nextInt();

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

            int n= input.nextInt();
            int x= input.nextInt();
            int a= input.nextInt();
            int b= input.nextInt();

            if (a==1 && b==n || b==1 && a==n){
                System.out.println(Math.abs(b-a));
            }else if (Math.abs(a-b)+x>=n){
                System.out.println(n-1);
            }else {
                System.out.println(Math.abs(a-b)+x);
            }
        }
    }
}
```

Lord Omkar has permitted you to enter the Holy Church of Omkar! To test your worthiness, Omkar gives you a password which you must interpret! A password is an array a of n positive integers. You apply the following operation to the array: pick any two adjacent numbers that are not equal to each other and replace them with their sum. Formally, choose an index i such that $1 \le i < n$ and $a_i \ne a_{i+1}$, delete both a_i and a_{i+1} from the array and put $a_i + a_{i+1}$ in their place.

For example, for array $[7,4,3,7]$ you can choose $i=2$ and the array will become $[7,4+3,7]=[7,7,7]$. Note that in this array you can't apply this operation anymore. Notice that one operation will decrease the size of the password by 1. What is the shortest possible length of the password after some number (possibly 0) of operations?

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 100)$. Description of the test cases follows.

The first line of each test case contains an integer n $(1 \le n \le 2 \cdot 10^{\wedge 5})$ — the length of the password. The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^{\wedge 9})$ — the initial contents of your password.

The sum of n over all test cases will not exceed $2 \cdot 10^{\wedge 5}$.

Output

For each password, print one integer: the shortest possible length of the password after some number of operations.

Answer (java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int num = input.nextInt();

        int sum = 0;
        for (int z = 0; z < num; z++) {

            int number = input.nextInt();
            int[] numbers = new int[number];

            for (int i = 0; i < number; i++) {
                numbers[i] = input.nextInt();
            }
            Arrays.sort(numbers);
            if (numbers[0] == numbers[number - 1]) {
                System.out.println(number);

            } else {
                System.out.println("1");
            }
        }
    }
}
```

You have been assigned to develop a filter for bad messages in the in-game chat. A message is a string S of length n, consisting of lowercase English letters and characters ')'. The message is bad if the number of characters ')' at the end of the string strictly greater than the number of remaining characters. For example, the string ")bc)))" has three parentheses at the end, three remaining characters, and is not considered bad.

Input

The first line contains the number of test cases $t (1 \le t \le 100)$. Description of the t test cases follows. The first line of each test case contains an integer n $(1 \le n \le 100)$. The second line of each test case contains a string S of length n, consisting of lowercase English letters and characters ')'.

Output

For each of t test cases, print "Yes" if the string is bad. Otherwise, print "No".

You can print each letter in any case (upper or lower).

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        String num1= input.nextLine();
        int num=Integer.parseInt(num1);
        for (int j=0;j<num;j++) {
            String num2= input.nextLine();
            int number=Integer.parseInt(num2);
            String word = input.nextLine();
            int sum = 0;
        for (int i = word.length() - 1; i >= 0;i--){
                if (word.charAt(i) == ')') {
                    sum = sum + 1;
                }
                if (word.charAt(i) != ')') {
                    break;
                }
            }
            if (number - sum >= sum) {
                System.out.println("NO");
                sum = 0;
            } else {
                System.out.println("YES");
                sum = 0;
            }
        }
    }
}
```

There are n students in a school class, the rating of the i-th student on Codehorses is a_i. You have to form a team consisting of k students ($1 \le k \le n$) such that the ratings of all team members are distinct.

If it is impossible to form a suitable team, print "NO" (without quotes). Otherwise print "YES", and then print k distinct numbers which should be the indices of students in the team you form. If there are multiple answers, print any of them.

Input

The first line contains two integers n and k ($1 \le k \le n \le 100$) — the number of students and the size of the team you have to form.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$), where a_i is the rating of i-th student.

Output

If it is impossible to form a suitable team, print "NO" (without quotes). Otherwise print "YES", and then print k distinct integers from 1 to n which should be the indices of students in the team you form. All the ratings of the students in the team should be distinct. You may print the indices in any order. If there are multiple answers, print any of them.

Assume that the students are numbered from 1 to n.

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static int removeduplicates(int a[], int
n)
    {
        if (n == 0 || n == 1) {
            return n;
        }

        // creating another array for only storing
        // the unique elements
        int[] temp = new int[n];
        int j = 0;

        for (int i = 0; i < n - 1; i++) {
            if (a[i] != a[i + 1]) {
                temp[j++] = a[i];
            }
        }

        temp[j++] = a[n - 1];

        // Changing the original array
        for (int i = 0; i < j; i++) {
            a[i] = temp[i];
        }

        return j;
    }


    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        int team= input.nextInt();
        int sum=0;
        int [] numbers=new int[number];
        int [] numbers1=new int[number];
        for (int i=0;i<number;i++){
            numbers[i]= input.nextInt();
            numbers1[i]=numbers[i];
        }

        Arrays.sort(numbers);
        int n=numbers.length;
        n=removeduplicates(numbers,n);
        int []arr=new int[n];
        for (int j=0;j<n;j++){
            arr[j]=numbers[j];
            sum=sum+1;
        }
        //System.out.println("arr:
"+Arrays.toString(arr));
        //System.out.println("numbers1:
"+Arrays.toString(numbers1));
        int count=0;
        if (sum>=team){
            System.out.println("YES");
            for (int o=0;o<n;o++){
                for (int p=0;p<number;p++){
                    if (arr[o]==numbers1[p] &&
count<team){
                        System.out.print((p+1)+" ");
                        arr[o]=-1;
                        numbers1[p]=0;
                        count=count+1;
                    }
                }
            }
        }else {
            System.out.println("NO");
        }
    }
}
```

You are given an integer n. You can perform any of the following operations with this number an arbitrary (possibly, zero) number of times:

1. Replace n with $n/2$ if n is divisible by 2;
2. Replace n with $2*n/3$ if n is divisible by 3;
3. Replace n with $4*n/5$ if n is divisible by 5.

For example, you can replace 30 with 15 using the first operation, with 20 using the second operation or with 24 using the third operation.

Your task is to find the minimum number of moves required to obtain 1 from n or say that it is impossible to do it. You have to answer q independent queries.

Input

The first line of the input contains one integer q ($1 \le q \le 1000$) — the number of queries. The next q lines contain the queries. For each query you are given the integer number n ($1 \le n \le 10$^18).

Output

Print the answer for each query on a new line. If it is impossible to obtain 1 from n, print -1. Otherwise, print the minimum number of moves required to do it.

Answer (java):

```
package com.company;
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int sum=0;
        boolean mines=false;

        for (int i=0;i<number;i++) {
            sum = 0;
            long numbers = input.nextLong();
            while (numbers != 1) {
                if (numbers % 5 == 0) {
                    numbers = (4 * numbers) / 5;
                    sum = sum + 1;
                } else if (numbers % 3 == 0) {
                    numbers = (numbers * 2) / 3;
                    sum = sum + 1;
                } else if (numbers % 2 == 0) {
                    numbers = numbers / 2;
                    sum = sum + 1;
                } else {
                    mines = true;
                    break;
                }
            }
            if (mines) {
                System.out.println("-1");
                mines=false;
            } else {
                System.out.println(sum);
            }
        }
    }
}
```

Petya has an array a consisting of n integers. He wants to remove duplicate (equal) elements.

Petya wants to leave only the rightmost entry (occurrence) for each element of the array. The relative order of the remaining unique elements should not be changed.

Input

The first line contains a single integer n ($1 \le n \le 50$) — the number of elements in Petya's array.

The following line contains a sequence $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 1000$) — the Petya's array.

Output

In the first line print integer x— the number of elements which will be left in Petya's array after he removed the duplicates.

In the second line print x integers separated with a space — Petya's array after he removed the duplicates. For each unique element only the rightmost entry should be left.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        int []numbers=new int[number];
        int sum=0;
        for (int i=0;i<number;i++){
            numbers[number-i-1]= input.nextInt();
        }
        for (int j=0;j<number-1;j++){
            for (int k=j+1;k<number;k++){
                if (numbers[j]==numbers[k]){
                    numbers[k]=0;
                }
            }
        }

//System.out.println(Arrays.toString(numbers));
        for (int k=0;k<number;k++){
            if (numbers[k]!=0){
                sum=sum+1;
            }
        }
        System.out.println(sum);

        for (int z=number-1;z>=0;z--){
            if (numbers[z]!=0){
                System.out.print(numbers[z]+" ");
            }
        }
    }
}
```

You are given four integers n, c_0, c_1 and h and a binary string s of length n. A binary string is a string consisting of characters 0 and 1. You can change any character of the string s (the string should be still binary after the change). You should pay h coins for each change.

After some changes (possibly zero) you want to buy the string. To buy the string you should buy all its characters. To buy the character 0 you should pay c_0 coins, to buy the character 1 you should pay c_1 coins. Find the minimum number of coins needed to buy the string.

Input

The first line contains a single integer t ($1 \le t \le 10$) — the number of test cases. Next $2t$ lines contain descriptions of test cases. The first line of the description of each test case contains four integers n, c_0, c_1, h ($1 \le n, c_0, c_1, h \le 1000$). The second line of the description of each test case contains the binary string s of length n.

Output

For each test case print a single integer — the minimum number of coins needed to buy the string.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner sc = new Scanner(System.in);

        int t = sc.nextInt();
        for (int tc = 0; tc < t; ++tc) {
            sc.nextInt();
            int c0 = sc.nextInt();
            int c1 = sc.nextInt();
            int h = sc.nextInt();
            String s = sc.next();

            System.out.println(solve(s, c0, c1, h));
        }

        sc.close();
    }

    static int solve(String s, int c0, int c1, int h)
{
        int result = 0;
        for (char ch : s.toCharArray()) {
            if (ch == '1') {
              result = result + Math.min(c1, h + c0);
            } else {
              result = result + Math.min(c0, c1 + h);
            }
        }
        return result;
    }
}
```

Two players are playing a game. First each of them writes an integer from 1 to 6, and then a dice is thrown. The player whose written number got closer to the number on the dice wins. If both payers have the same difference, it's a draw.

The first player wrote number a, the second player wrote number b. How many ways to throw a dice are there, at which the first player wins, or there is a draw, or the second player wins?

Input

The single line contains two integers a and b ($1 \le a, b \le 6$) — the numbers written on the paper by the first and second player, correspondingly.

Output

Print three integers: the number of ways to throw the dice at which the first player wins, the game ends with a draw or the second player wins, correspondingly.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Locale;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int playerA = input.nextInt();
        int playerB = input.nextInt();

        int sum1=0;
        int sum2=0;
        int draw=0;

        for (int i=1;i<=6;i++){
            int resultA=Math.abs(playerA-i);
            int resultB=Math.abs(playerB-i);

            if (resultA>resultB){
                sum1++;
            }else if (resultB>resultA){
                sum2++;
            }else {
                draw++;
            }
        }

        System.out.println(sum2+" "+draw+" "+sum1);

    }
}
```

Michael is accused of violating the social distancing rules and creating a risk of spreading coronavirus. He is now sent to prison. Luckily, Michael knows exactly what the prison looks like from the inside, especially since it's very simple.

The prison can be represented as a rectangle $a \times b$ which is divided into ab cells, each representing a prison cell, common sides being the walls between cells, and sides on the perimeter being the walls leading to freedom. Before sentencing, Michael can ask his friends among the prison employees to make (very well hidden) holes in some of the walls (including walls between cells and the outermost walls). Michael wants to be able to get out of the prison after this, no matter which cell he is placed in. However, he also wants to break as few walls as possible.

Your task is to find out the smallest number of walls to be broken so that there is a path to the outside from every cell after this.

Input

The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases. Each of the following t lines contains two integers a and b $(1 \le a, b \le 100)$, representing a corresponding test case.

Output

For each test case print the single integer on a separate line — the answer to the problem.

Answer (java):

```
package com.company;
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        for (int j=0;j<number;j++) {
            int a=input.nextInt();
            int b= input.nextInt();
            System.out.println(a*b);

        }


    }
}
```

Mishka started participating in a programming contest. There are n problems in the contest. Mishka's problem-solving skill is equal to k. Mishka arranges all problems from the contest into a list. Because of his weird principles, Mishka only solves problems from one of the ends of the list. Every time, he chooses which end (left or right) he will solve the next problem from. Thus, each problem Mishka solves is either the leftmost or the rightmost problem in the list. Mishka cannot solve a problem with difficulty greater than k. When Mishka solves the problem, it disappears from the list, so the length of the list decreases by 1. Mishka stops when he is unable to solve any problem from any end of the list. How many problems can Mishka solve?

Input

The first line of input contains two integers n and k ($1 \le n,k \le 100$) — the number of problems in the contest and Mishka's problem-solving skill. The second line of input contains n integers $a_1,a_2,\dots,a_n$ ($1 \le a_i \le 100$), where a_i is the difficulty of the i-th problem. The problems are given in order from the leftmost to the rightmost in the list.

Output

Print one integer — the maximum number of problems Mishka can solve.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int hardship = input.nextInt();
        int []arr=new int[number];
        int sum=0;
        for (int i=0;i<number;i++) {
            arr[i]=input.nextInt();
        }

        int k=0;
        while (arr[k]<=hardship){
            if (k==number-1){
                break;
            }
            sum=sum+1;
            k++;
        }

        int j=0;
        while (arr[number-j-1]<=hardship){
            if (j==number-1){
                break;
            }
            sum=sum+1;
            j++;

        }
        Arrays.sort(arr);
        if (arr[0]<=hardship && arr[number-
1]<=hardship){
            System.out.println(number);
        }else {
            System.out.println(sum);
        }
    }
}
```

Two players play a simple game. Each player is provided with a box with balls. First player's box contains exactly n_1 balls and second player's box contains exactly n_2 balls. In one move first player can take from 1 to k_1 balls from his box and throw them away. Similarly, the second player can take from 1 to k_2 balls from his box in his move. Players alternate turns and the first player starts the game. The one who can't make a move loses. Your task is to determine who wins if both players play optimally.

Input

The first line contains four integers n_1, n_2, k_1, k_2. All numbers in the input are from 1 to 50.

This problem doesn't have subproblems. You will get 3 points for the correct submission.

Output

Output "First" if the first player wins and "Second" otherwise.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int n1=input.nextInt();
        int n2= input.nextInt();
        int k1=input.nextInt();
        int k2= input.nextInt();
        int times1=n1/k1;
        int times2=n2/k2;

        if (n1 > n2){
            System.out.println("First");
        }else {
            System.out.println("Second");
        }
    }
}
```

You are given a set of n segments on the axis Ox, each segment has integer endpoints between 1 and m inclusive. Segments may intersect, overlap or even coincide with each other. Each segment is characterized by two integers l_i and r_i ($1 \le l_i \le r_i \le m$) coordinates of the left and of the right endpoints.

Consider all integer points between 1 and m inclusive. Your task is to print all such points that don't belong to any segment. The point x belongs to the segment $[l;r]$ if and only if $l \le x \le r$.

Input

The first line of the input contains two integers n and m ($1 \le n,m \le 100$) — the number of segments and the upper bound for coordinates.

The next n lines contain two integers each l_i and r_i ($1 \le l_i \le r_i \le m$) — the endpoints of the i-th segment. Segments may intersect, overlap or even coincide with each other. Note, it is possible that $l_i = r_i$, i.e. a segment can degenerate to a point.

Output

In the first line print one integer k the number of points that don't belong to any segment. In the second line print exactly k integers in any order — the points that don't belong to any segment. All points you print should be distinct.

If there are no such points at all, print a single integer 0 in the first line and either leave the second line empty or do not print it at all.

Answer (java):

```java
package com.company;
import java.awt.image.AreaAveragingScaleFilter;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int t = input.nextInt();
        int till = input.nextInt();
        int []numbers=new int[till];
        for (int j=1;j<=till;j++){
            numbers[j-1]=j;
        }
        for (int i=0;i<t;i++){
            int from=input.nextInt();
            int to= input.nextInt();

            for (int z=from-1;z<to;z++){
                numbers[z]=0;
            }

        }
        int sum=0;
        for (int k=0;k<numbers.length;k++){
            if (numbers[k]!=0){
                sum++;
            }
        }
        System.out.println(sum);

        for (int k2=0;k2<numbers.length;k2++){
            if (numbers[k2]!=0){
                System.out.print(numbers[k2]+" ");
            }
        }
    }
}
```

n students are taking an exam. The highest possible score at this exam is m. Let a_i be the score of the i-th student. You have access to the school database which stores the results of all students.You can change each student's score as long as the following conditions are satisfied:

- All scores are integers $0 \le a_i \le m$ The average score of the class doesn't change.

You are student 1 and you would like to maximize your own score.

Find the highest possible score you can assign to yourself such that all conditions are satisfied.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 200)$. The description of the test cases follows. The first line of each test case contains two integers n and m $(1 \le n \le 10^3,\ 1 \le m \le 10^5)$ the number of students and the highest possible score respectively. The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le m)$ — scores of the students.

Output

For each testcase, output one integer — the highest possible score you can assign to yourself such that both conditions are satisfied._

Answer(java):

```java
package com.company;
import java.awt.image.AreaAveragingScaleFilter;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int num= input.nextInt();

        for (int j=0;j<num;j++) {
            int number = input.nextInt();
            int m = input.nextInt();
            int[] numbers = new int[number];
            int sum = 0;
            for (int i = 0; i < number; i++) {
                numbers[i] = input.nextInt();
                sum = sum + numbers[i];
            }

            System.out.println(Math.min(sum, m));
        }
    }
}
```

It's one more school day now. Sasha doesn't like classes and is always bored at them. So, each day he invents some game and plays in it alone or with friends.

Today he invented one simple game to play with Lena, with whom he shares a desk. The rules are simple. Sasha draws n sticks in a row. After that the players take turns crossing out exactly k sticks from left or right in each turn. Sasha moves first, because he is the inventor of the game. If there are less than k sticks on the paper before some turn, the game ends. Sasha wins if he makes strictly more moves than Lena. Sasha wants to know the result of the game before playing, you are to help him.

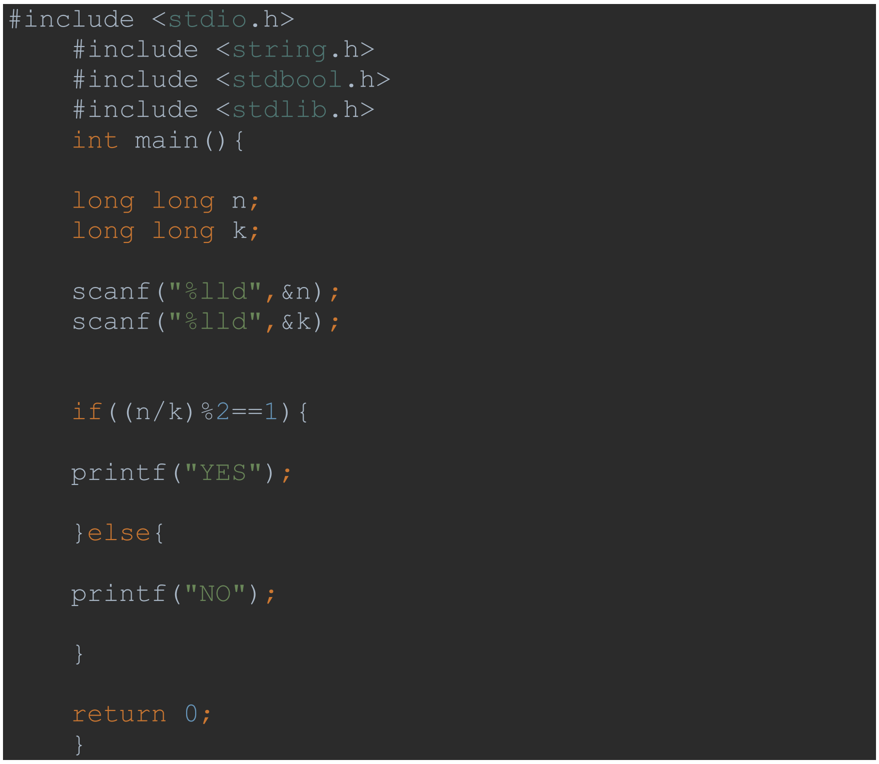
Input

The first line contains two integers n and k ($1 \le n, k \le 10^{18}$, $k \le n$) — the number of sticks drawn by Sasha and the number k the number of sticks to be crossed out on each turn.

Output

If Sasha wins, print "YES" (without quotes), otherwise print "NO" (without quotes). You can print each letter in arbitrary case (upper of lower).

Answer (C):



```
#include <stdio.h>
    #include <string.h>
    #include <stdbool.h>
    #include <stdlib.h>
    int main(){

    long long n;
    long long k;

    scanf("%lld",&n);
    scanf("%lld",&k);


    if((n/k)%2==1){

    printf("YES");

    }else{

    printf("NO");

    }

    return 0;
    }
```

Polycarp wants to cook a soup. To do it, he needs to buy exactly $n$liters of water. There are only two types of water bottles in the nearby shop 1-liter bottles and 2-liter bottles. There are infinitely many bottles of these two types in the shop.

The bottle of the first type costs a burles and the bottle of the second type costs b burles correspondingly.

Polycarp wants to spend as few money as possible. Your task is to find the minimum amount of money (in burles) Polycarp needs to buy exactly n liters of water in the nearby shop if the bottle of the first type costs a burles and the bottle of the second type costs b burles. You also have to answer q independent queries.

Input

The first line of the input contains one integer $q(1 \le q \le 500)$ the number of queries. The next q lines contain queries. The i-th query is given as three space-separated integers n_i, a_i and b_i $(1 \le n_i \le 10_{12}, 1 \le a_i, b_i \le 1000)$ — how many liters Polycarp needs in the i-th query, the cost (in burles) of the bottle of the first type in the i-th query and the cost (in burles) of the bottle of the second type in the i-th query, respectively.

Output

Print q integers. The i-th integer should be equal to the minimum amount of money (in burles) Polycarp needs to buy exactly n_i liters of water in the nearby shop if the bottle of the first type costs a_i burles and the bottle of the second type costs b_i burles.

Answer(java):

```java
package com.company;
import java.awt.image.AreaAveragingScaleFilter;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]){

        Scanner input=new Scanner(System.in);
        int nnumber= input.nextInt();
        for (int i=0;i<nnumber;i++){
            long n= input.nextLong();
            long a= input.nextLong();
            long b= input.nextLong();

            if (2*a>b){
                if (n%2==0){
                    System.out.println((n/2)*b);
                }else {
                  System.out.println(((n-1)/2*b+a));
                }
            }else{
                System.out.println(a*n);
            }
        }
    }
}
```

You have unlimited number of coins with values $1,2,\ldots,n$. You want to select some set of coins having the total value of S. It is allowed to have multiple coins with the same value in the set. What is the minimum number of coins required to get sum S?

Input

The only line of the input contains two integers n and S ($1 \le n \le 100000$, $1 \le S \le 10$^9)

Output

Print exactly one integer — the minimum number of coins required to obtain sum S.

Answer(java):

```
package com.company;
import java.awt.image.AreaAveragingScaleFilter;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int n= input.nextInt();
        int price= input.nextInt();

        if (n>=price){
            System.out.println("1");
        }else {
            if (price % n == 0) {
                System.out.println(price / n);
            } else {
                int result = price / n;
                System.out.println(result + 1);
            }
        }




    }
}
```

Tomorrow is a difficult day for Polycarp: he has to attend a lectures and b practical classes at the university! Since Polycarp is a diligent student, he is going to attend all of them.

While preparing for the university, Polycarp wonders whether he can take enough writing implements to write all of the lectures and draw everything he has to during all of the practical classes. Polycarp writes lectures using a pen (he can't use a pencil to write lectures!); he can write down c lectures using one pen, and after that it runs out of ink. During practical classes Polycarp draws blueprints with a pencil (he can't use a pen to draw blueprints!); one pencil is enough to draw all blueprints during d practical classes, after which it is unusable. Polycarp's pencilcase can hold no more than k writing implements, so if Polycarp wants to take x pens and y pencils, they will fit in the pencilcase if and only if $x+y\le k$.

Now Polycarp wants to know how many pens and pencils should he take. Help him to determine it, or tell that his pencilcase doesn't have enough room for all the implements he needs tomorrow!

Note that you don't have to minimize the number of writing implements (though their total number must not exceed k).

Input

The first line of the input contains one integer t ($1\le t\le 100$) the number of test cases in the input. Then the test cases follow. Each test case is described by one line containing five integers a, b, c, d and k, separated by spaces ($1\le a,b,c,d,k\le 100$) the number of lectures Polycarp has to attend, the number of practical classes Polycarp has to attend, the number of lectures which can be written down using one pen, the number of practical classes for which one pencil is enough, and the number of writing implements that can fit into Polycarp's pencilcase, respectively. In hacks it is allowed to use only one test case in the input, so $t=1$ should be satisfied.

Output

For each test case, print the answer as follows: If the pencilcase can't hold enough writing implements to use them during all lectures and practical classes, print one integer -1. Otherwise, print two non-negative integers x and y — the number of pens and pencils Polycarp should put in his pencilcase. If there are multiple answers, print any of them. Note that you don't have to minimize the number of writing implements (though their total number must not exceed k).

Answer(java):

```
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        for (int i=0;i<number;i++){
            int lectures= input.nextInt();
            int oClass= input.nextInt();
            int penPerLecture = input.nextInt();
            int classPerPencil= input.nextInt();
            int pencase= input.nextInt();
            int pen=0;
            for (int j=1;j<=lectures;j++){
                if (penPerLecture*j>=lectures){
                    pen=j;
                    break;
                }
            }
            for (int k=1;k<=oClass;k++){
                if (classPerPencil*k>=oClass){
                    classPerPencil=k;
                    break;
                }
            }
            pencase=pencase-pen-classPerPencil;
            pen=pen+(pencase);
            if (pencase>=0) {
                System.out.print(pen + " " +
classPerPencil);
                System.out.println();
            }else {
                System.out.println("-1");
            }
        }
    }
}
```

You are given a string s, consisting of brackets of two types: '(', ')', '[' and ']'.A string is called a regular bracket sequence (RBS) if it's of one of the following types:

- empty string;
- '(' + RBS + ')';
- '[' + RBS + ']';
- RBS + RBS.

where plus is a concatenation of two strings. In one move you can choose a non-empty subsequence of the string s (not necessarily consecutive) that is an RBS, remove it from the string and concatenate the remaining parts without changing the order. What is the maximum number of moves you can perform?

Input

The first line contains a single integer t ($1 \le t \le 1000$) the number of testcases. Each of the next t lines contains a non-empty string, consisting only of characters '(', ')', '[' and ']'. The total length of the strings over all testcases doesn't exceed $2\cdot10$^5.

Output

For each testcase print a single integer — the maximum number of moves you can perform on a given string s.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        String num= input.nextLine();
        int number =Integer.parseInt(num);
        int r=0;
        int s=0;
        int ans=0;
        for (int j=0;j<number;j++) {
            String word= input.nextLine();

            for (int i=0;i<word.length();i++){
                if (word.charAt(i)=='('){
                    r++;
                }else if (word.charAt(i)=='['){
                    s++;
               }else if (word.charAt(i)==')' && r>0){
                    r--;
                    ans++;
               }else if (word.charAt(i)==']' && s>0){
                    s--;
                    ans++;
                }
            }
            System.out.println(ans);
            ans=0;
            r=0;
            s=0;
        }
    }
}
```

Masha has three sticks of length a, b and c

centimeters respectively. In one minute Masha can pick one arbitrary stick and increase its length by one centimeter. She is not allowed to break sticks.

What is the minimum number of minutes she needs to spend increasing the stick's length in order to be able to assemble a triangle of positive area. Sticks should be used as triangle's sides (one stick for one side) and their endpoints should be located at triangle's vertices.

Input

The only line contains tree integers a, b and c ($1 \le a,b,c \le 100$) the lengths of sticks Masha possesses.

Output

Print a single integer — the minimum number of minutes that Masha needs to spend in order to be able to make the triangle of positive area from her sticks.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        int[]triangle=new int[3];
        triangle[0]= input.nextInt();
        triangle[1]= input.nextInt();
        triangle[2]= input.nextInt();
        Arrays.sort(triangle);

        if (triangle[0]+triangle[1]>triangle[2]){
            System.out.println("0");
        }else {
            System.out.println(triangle[2]-
triangle[0]-triangle[1]+1);
        }
    }
}
```

You are given an array a consisting of n integers. Initially all elements of a are either 0 or 1. You need to process q queries of two kinds:

- 1 x : Assign to a_x the value $1-a_x$.
- 2 k : Print the k-th largest value of the array.

As a reminder, k-th largest value of the array b is defined as following:

- Sort the array in the non-increasing order, return k-th element from it.

For example, the second largest element in array $[0,1,0,1]$ is 1, as after sorting in non-increasing order it becomes $[1,1,0,0]$, and the second element in this array is equal to 1.

Input

The first line contains two integers n and q ($1\le n,q\le 10$^5) — the length of the given array and the number of queries. The second line contains n integers $a_1,a_2,a_3,\dots,a_n$ ($0\le a_i\le 1$) elements of the initial array.

Each of the following q lines contains two integers. The first integer is t ($1\le t\le 2$) the type of query.

- If $t=1$the second integer is x ($1\le x\le n$) — the position of the modified number. You have to assign to a_x the value $1-a_x$.

- If $t=2$ the second integer is k ($1 \le k \le n$) — you need to print the k-th largest value of the array.

It's guaranteed that there will be at least one query of the second type (satisfying $t=2$).

Output

For each query of the second type, print a single integer — the answer to the query.

Answer(java):

```java
package com.company;
import java.util.Scanner;
public class Main {
    public static void main(String args[]){
        Scanner input=new Scanner(System.in);
        int length= input.nextInt();
        int query = input.nextInt();
        int[]numbers=new int[length];
        int max=0;
        for (int i=0;i<length;i++){
            numbers[i]= input.nextInt();
            if (numbers[i]==1){
                max++;
            }
        }

        for (int j=0;j<query;j++){
            int t= input.nextInt();
            int x= input.nextInt();

            if (t==1){
                if (numbers[x-1]==0){
                    numbers[x-1]=1;
                    max++;
                }else if (numbers[x-1]==1){
                    numbers[x-1]=0;
                    max--;
                }
            }else if (t==2){
                if (max>=x){
                    System.out.println("1");
                }else {
                    System.out.println("0");
                }
            }
        }
    }
}
```

Alice has a string s. She really likes the letter "a". She calls a string good if strictly more than half of the characters in that string are "a"s. For example "aaabb", "axaa" are good strings, and "baca", "awwwa", "" (empty string) are not. Alice can erase some characters from her string s. She would like to know what is the longest string remaining after erasing some characters (possibly zero) to get a good string. It is guaranteed that the string has at least one "a" in it, so the answer always exists.

Input

The first line contains a string s ($1 \le |s| \le 50$) consisting of lowercase English letters. It is guaranteed that there is at least one "a" in s.

Output

Print a single integer, the length of the longest good string that Alice can get after erasing some characters from s.

Answer (java):

```java
//package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        String word = input.nextLine();
        int sum = 0;
        for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) == 'a') {
                sum = sum + 1;
            }
        }

System.out.println(Math.min(word.length(),2*sum-1));
    }
}
```

A telephone number is a sequence of exactly 11 digits, where the first digit is 8. For example, the sequence 80011223388 is a telephone number, but the sequences 70011223388 and 80000011223388 are not. You are given a string s of length n, consisting of digits. In one operation you can delete any character from string s. For example, it is possible to obtain strings 112, 111 or 121 from string 1121. You need to determine whether there is such a sequence of operations (possibly empty), after which the string s becomes a telephone number.

Input

The first line contains one integer $t (1 \le t \le 100)$ the number of test cases. The first line of each test case contains one integer n $(1 \le n \le 100)$ — the length of string s. The second line of each test case contains the string s $(|s| = n)$ consisting of digits.

Output

For each test print one line. If there is a sequence of operations, after which s becomes a telephone number, print YES. Otherwise, print NO.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String num1 = input.nextLine();
        int num = Integer.parseInt(num1);

        for (int i = 0; i < num; i++) {
            String num2 = input.nextLine();
            int number = Integer.parseInt(num2);
            String numbers = input.nextLine();


            if (numbers.length() -
numbers.indexOf('8') >= 11 && numbers.contains("8"))
{
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }

        }
    }
}
```

There are n cards (n is even) in the deck. Each card has a positive integer written on it. $n / 2$ people will play new card game. At the beginning of the game each player gets two cards, each card is given to exactly one player.

Find the way to distribute cards such that the sum of values written of the cards will be equal for each player. It is guaranteed that it is always possible.

Input

The first line of the input contains integer n ($2 \leq n \leq 100$) — the number of cards in the deck. It is guaranteed that n is even.

The second line contains the sequence of n positive integers $a_1, a_2, ..., a_n$ ($1 \leq a_i \leq 100$), where a_i is equal to the number written on the i-th card.

Output

Print $n / 2$ pairs of integers, the i-th pair denote the cards that should be given to the i-th player. Each card should be given to exactly one player. Cards are numbered in the order they appear in the input.

It is guaranteed that solution exists. If there are several correct answers, you are allowed to print any of them.

Answer(java):

```java
package com.company;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;

public class main {

    public static int find(int[] a, int target)
    {
        for (int i = 0; i < a.length; i++)
        {
            if (a[i] == target) {
                a[i]=0;
                return i;
            }
        }

        return -1;
    }

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int[] arr = new int[number];
        int[] copyArr = new int[number];

        for (int i = 0; i < number; i++) {
            int numbers = input.nextInt();
            arr[i] = numbers;
            copyArr[i] = numbers;
        }
        Arrays.sort(arr);
        for (int j=0;j<(number/2);j++){
            int first = arr[j];
            int last = arr[number-1-j];
            int la1=find(copyArr,first);
            int la2=find(copyArr,last);
        System.out.println(((la1)+1)+" "+((la2)+1));
        }
    }
}
```

We just discovered a new data structure in our research group: a suffix three! It's very useful for natural language processing. Given three languages and three suffixes, a suffix three can determine which language a sentence is written in. It's super simple, 100% accurate, and doesn't involve advanced machine learning algorithms.

Let us tell you how it works.

- If a sentence ends with "po" the language is Filipino.
- If a sentence ends with "desu" or "masu" the language is Japanese.
- If a sentence ends with "mnida" the language is Korean.

Given this, we need you to implement a suffix three that can differentiate Filipino, Japanese, and Korean. Oh, did I say three suffixes? I meant four.

Input

The first line of input contains a single integer t ($1 \leq t \leq 30$) denoting the number of test cases. The next lines contain descriptions of the test cases.

Each test case consists of a single line containing a single string denoting the sentence. Spaces are represented as underscores (the symbol "_") for ease of reading. The sentence has at least $1$and at most 1000 characters, and consists only of lowercase English letters and underscores. The sentence has no leading or trailing underscores and no two consecutive underscores. It is guaranteed that the sentence ends with one of the four suffixes mentioned above.

Output

For each test case, print a single line containing either "FILIPINO", "JAPANESE", or "KOREAN" (all in uppercase, without quotes), depending on the detected language.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]){

        Scanner input=new Scanner(System.in);
        String number1= input.nextLine();
        int number= Integer.parseInt(number1);
        for (int i=0;i<number;i++){
            String word= input.nextLine();

            if (word.charAt(word.length()-1)=='o'){
                System.out.println("FILIPINO");
        }else if (word.charAt(word.length()-1)=='u'){
                System.out.println("JAPANESE");
        }else if (word.charAt(word.length()-1)=='a'){
                System.out.println("KOREAN");
            }
        }
    }
}
```

Nikolay has a lemons, b apples and c pears. He decided to cook a compote. According to the recipe the fruits should be in the ratio 1: 2: 4. It means that for each lemon in the compote should be exactly 2 apples and exactly 4 pears. You can't crumble up, break up or cut these fruits into pieces. These fruits — lemons, apples and pears — should be put in the compote as whole fruits.

Your task is to determine the maximum total number of lemons, apples and pears from which Nikolay can cook the compote. It is possible that Nikolay can't use any fruits, in this case print 0.

Input

The first line contains the positive integer a ($1 \le a \le 1000$) — the number of lemons Nikolay has.

The second line contains the positive integer b ($1 \le b \le 1000$) — the number of apples Nikolay has.

The third line contains the positive integer c ($1 \le c \le 1000$) — the number of pears Nikolay has.

Output

Print the maximum total number of lemons, apples and pears from which Nikolay can cook the compote.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Vector;

public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int a= input.nextInt();
        int b= input.nextInt();
        int c= input.nextInt();

        if (a<1 || b<2 || c<4){
            System.out.println("0");
        }else {
            int min=Math.min(a,b/2);
            int min2=Math.min(c/4,min);
            int result=min2*7;
            System.out.println(result);
        }
    }
}
```

Tokitsukaze is one of the characters in the game "Kantai Collection". In this game, every character has a common attribute — health points, shortened to HP.

In general, different values of HP are grouped into 4

categories:

- Category A

if HP is in the form of $(4n+1)$, that is, when divided by 4, the remainder is 1

- ;
- Category B if HP is in the form of $(4n+3)$, that is, when divided by 4, the remainder is 3
- ;
- Category C if HP is in the form of $(4n+2)$, that is, when divided by 4, the remainder is 2
- ;
- Category D if HP is in the form of $4n$, that is, when divided by 4, the remainder is 0

- .

The above-mentioned n

can be any integer.

These 4

categories ordered from highest to lowest as $A>B>C>D$, which means category A is the highest and category D

is the lowest.

While playing the game, players can increase the HP of the character. Now, Tokitsukaze wants you to increase her HP by at most 2(that is, either by 0, 1 or 2). How much should she increase her HP so that it has the highestpossible category?

Input

The only line contains a single integer x ($30 \le x \le 100$) the value Tokitsukaze's HP currently.

Output

Print an integer a ($0 \le a \le 2$) and an uppercase letter b ($b \in \{A,B,C,D\}$), representing that the best way is to increase her HP by a, and then the category becomes b.

Note that the output characters are case-sensitive.

Answer (java):

```java
package com.company;
import java.util.Scanner;

class main {

    public static void main(String[]args){
        Scanner input = new Scanner(System.in);

        int number = input.nextInt();

        if (number%4==0){
            System.out.println("1 A");
        }else if (number%4==1){
            System.out.println("0 A");
        }else if (number%4==2){
            System.out.println("1 B");
        }else {
            System.out.println("2 A");
        }

    }
}
```

2 900 SCORE PROBLEMS

You are given n numbers $a_1, a_2, \ldots, a_n$. With a cost of one coin you can perform the following operation:
Choose one of these numbers and add or subtract 1 from it.
In particular, we can apply this operation to the same number several times. We want to make the product of all these numbers equal to 1, in other words, we want $a_1 \cdot a_2 \ldots \cdot a_n = 1$.

For example, for $n=3$ and numbers $[1,-3,0]$ we can make product equal to 1 in 3 coins: add 1 to second element, add 1 to second element again, subtract 1 from third element, so that array becomes $[1,-1,-1]$. And $1 \cdot (-1) \cdot (-1) = 1$. What is the minimum cost we will have to pay to do that?

Input

The first line contains a single integer n ($1 \le n \le 10$^5) the number of numbers. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10_9 \le a_i \le 10$^9) the numbers.

Output

Output a single number — the minimal number of coins you need to pay to make the product equal to 1.

Answer(java):

```java
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        long sum=0;
        long sign=0;
        boolean zero=false;
        for (int i=0;i<number;i++){
            long numbers= input.nextLong();
            if (numbers==0){
                zero=true;
            }
            if (numbers==-1){
                sign=sign+1;
            }
            if (numbers!=1 && numbers!=-1){
                if (numbers<0){
                    sum=sum+(Math.abs(numbers+1));
                    sign = sign + 1;
                }else{
                    sum=sum+Math.abs(numbers-1);
                }
            }
        }
        if (sign%2==1 && !zero){
            System.out.println(sum+2);
        }else {
            System.out.println(Math.abs(sum));
        }
    }
}
```

Duff is addicted to meat! Malek wants to keep her happy for n days. In order to be happy in i-th day, she needs to eat exactly a_i kilograms of meat.

There is a big shop uptown and Malek wants to buy meat for her from there. In i-th day, they sell meat for p_i dollars per kilogram. Malek knows all numbers $a_1, ..., a_n$ and $p_1, ..., p_n$. In each day, he can buy arbitrary amount of meat, also he can keep some meat he has for the future.

Malek is a little tired from cooking meat, so he asked for your help. Help him to minimize the total money he spends to keep Duff happy for n days.

Input

The first line of input contains integer n ($1 \le n \le 10^5$), the number of days.

In the next n lines, i-th line contains two integers a_i and p_i ($1 \le a_i, p_i \le 100$), the amount of meat Duff needs and the cost of meat in that day.

Output

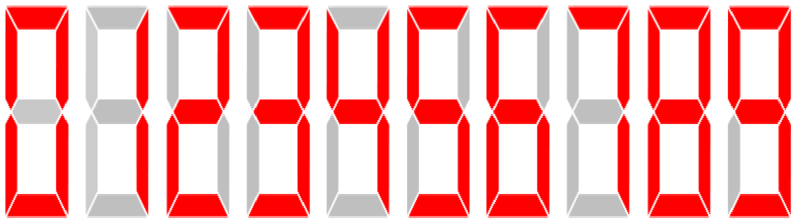
Print the minimum money needed to keep Duff happy for n days, in one line.

Answer(java):

```java
package com.company;
import java.util.*;
import java.io.*;

public class Main {

    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int ans=0;
        int mn=200;

        for (int i=0;i<n;i++){
            int a= input.nextInt();
            int p= input.nextInt();
            mn = Math.min(mn,p);
            ans=ans+a*mn;
        }
        System.out.println(ans);

    }
}
```

You have a large electronic screen which can display up to 998244353 decimal digits. The digits are displayed in the same way as on different electronic alarm clocks: each place for a digit consists of 7 segments which can be turned on and off to compose different digits. The following picture describes how you can display all 10

decimal digits:



As you can see, different digits may require different number of segments to be turned on. For example, if you want to display 1, you have to turn on 2 segments of the screen, and if you want to display 8, all 7 segments of some place to display a digit should be turned on.

You want to display a really large integer on the screen. Unfortunately, the screen is bugged: no more than n segments can be turned on simultaneously. So now you wonder what is the greatest integer that can be displayed by turning on no more than n segments. Your program should be able to process t different test cases.

Input

The first line contains one integer t ($1 \le t \le 100$) the number of test cases in the input.

Then the test cases follow, each of them is represented by a separate line containing one integer n ($2 \le n \le 10$^5) the maximum number of segments that can be turned on in the corresponding testcase.

It is guaranteed that the sum of n over all test cases in the input does not exceed 10^5.

Output

For each test case, print the greatest integer that can be displayed by turning on no more than n segments of the screen. Note that the answer may not fit in the standard 32-bit or 64-bit integral data type.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        for(int i=0;i<number;i++){
            int numbers= input.nextInt();
            if (numbers % 2 == 1) {
                System.out.print("7");
                for (int j=0;j<(numbers-3)/2;j++){
                    System.out.print("1");
                }
                System.out.println(" ");
            }else {
                for (int j=0;j<numbers/2;j++){
                    System.out.print("1");
                }
                System.out.println(" ");
            }

        }
    }
}
```

Having written another programming contest, three Rabbits decided to grab some lunch. The coach gave the team exactly k time units for the lunch break.

The Rabbits have a list of n restaurants to lunch in: the i-th restaurant is characterized by two integers f_i and t_i. Value t_i shows the time the Rabbits need to lunch in the i-th restaurant. If time t_i exceeds the time k that the coach has given for the lunch break, then the Rabbits' joy from lunching in this restaurant will equal $f_i - (t_i - k)$. Otherwise, the Rabbits get exactly f_i units of joy.

Your task is to find the value of the maximum joy the Rabbits can get from the lunch, depending on the restaurant. The Rabbits must choose exactly one restaurant to lunch in. Note that the joy value isn't necessarily a positive value.

Input

The first line contains two space-separated integers — n $(1 \le n \le 10^4)$ and k $(1 \le k \le 10^9)$ — the number of restaurants in the Rabbits' list and the time the coach has given them to lunch, correspondingly. Each of the next n lines contains two space-separated integers — f_i $(1 \le f_i \le 10^9)$ and t_i $(1 \le t_i \le 10^9)$ — the characteristics of the i-th restaurant.

Output

In a single line print a single integer — the maximum joy value that the Rabbits will get from the lunch.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int k = input.nextInt();
        int []arr=new int[n];
        for (int i=0;i<n;i++){
            int f= input.nextInt();
            int t= input.nextInt();

            if (t>k){
                arr[i]=f-(t-k);
            }else{
                arr[i]=f;
            }
        }
        Arrays.sort(arr);
        System.out.println(arr[n-1]);
    }
}
```

You are given two integers x and y (it is guaranteed that $x>y$). You may choose any prime integer p and subtract it any number of times from x. Is it possible to make x equal to y? Recall that a prime number is a positive integer that has exactly two positive divisors: 1 and this integer itself. The sequence of prime numbers starts with 2, 3, 5, 7, 11. Your program should solve t independent test cases.

Input

The first line contains one integer t ($1 \le t \le 1000$) the number of test cases.

Then t lines follow, each describing a test case. Each line contains two integers x and y ($1 \le y < x \le 10$^18).

Output

For each test case, print YES if it is possible to choose a prime number p and subtract it any number of times from x so that x becomes equal to y. Otherwise, print NO. You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes, and YES will all be recognized as positive answer).

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int n= input.nextInt();
        for (int i=0;i<n;i++) {

            long number = input.nextLong();
            long numbers = input.nextLong();
            if (number - numbers == 1) {
                System.out.println("NO");
            } else {
                System.out.println("YES");
            }
        }
    }
}
```

Kolya is going to make fresh orange juice. He has n oranges of sizes $a_1, a_2, ..., a_n$. Kolya will put them in the juicer in the fixed order, starting with orange of size a_1, then orange of size a_2 and so on. To be put in the juicer the orange must have size not exceeding b, so if Kolya sees an orange that is strictly greater he throws it away and continues with the next one.

The juicer has a special section to collect waste. It overflows if Kolya squeezes oranges of the total size strictly greater than d. When it happens Kolya empties the waste section (even if there are no more oranges) and continues to squeeze the juice. How many times will he have to empty the waste section?

Input

The first line of the input contains three integers n, b and d ($1 \le n \le 100\,000$, $1 \le b \le d \le 1\,000\,000$) — the number of oranges, the maximum size of the orange that fits in the juicer and the value d, which determines the condition when the waste section should be emptied.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 1\,000\,000$) — sizes of the oranges listed in the order Kolya is going to try to put them in the juicer.

Output

Print one integer — the number of times Kolya will have to empty the waste section.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int n=input.nextInt();
        int b= input.nextInt();
        int d=input.nextInt();
        int sum=0;
        int empty=0;
        for (int i=0;i<n;i++) {
            int a = input.nextInt();
            if (a > b) {
                continue;
            } else {
                sum = sum + a;
                if (sum > d) {
                    empty = empty + 1;
                    sum = 0;
                }
            }
        }
        System.out.println(empty);
    }
}
```

"Night gathers, and now my watch begins. It shall not end until my death. I shall take no wife, hold no lands, father no children. I shall wear no crowns and win no glory. I shall live and die at my post. I am the sword in the darkness. I am the watcher on the walls. I am the shield that guards the realms of men. I pledge my life and honor to the Night's Watch, for this night and all the nights to come." The Night's Watch oath.

With that begins the watch of Jon Snow. He is assigned the task to support the stewards.

This time he has n stewards with him whom he has to provide support. Each steward has his own strength. Jon Snow likes to support a steward only if there exists at least one steward who has strength strictly less than him and at least one steward who has strength strictly greater than him.

Can you find how many stewards will Jon support?

Input

First line consists of a single integer n ($1 \le n \le 10^5$) — the number of stewards with Jon Snow.

Second line consists of n space separated integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^9$) representing the values assigned to the stewards.

Output

Output a single integer representing the number of stewards which Jon will feed.

Answer(java):

```java
package com.company;
import java.awt.image.AreaAveragingScaleFilter;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        if (number<=2){
            System.out.println("0");
        }else {
            int[] numbers = new int[number];
            for (int i = 0; i < number; i++) {
                numbers[i] = input.nextInt();
            }
            Arrays.sort(numbers);
            int first=numbers[0];
            int last=numbers[number-1];

            int sum=0;
            for (int j=0;j<number;j++){
         if (numbers[j]!=first && numbers[j]!=last){
                    sum++;
                }
            }
            System.out.println(sum);
        }
    }
}
```

Three friends are going to meet each other. Initially, the first friend stays at the position $x=a$, the second friend stays at the position $x=b$ and the third friend stays at the position $x=c$ on the coordinate axis Ox. In one minute each friend independently from other friends can change the position x by 1 to the left or by 1 to the right (i.e. set $x:=x-1$ or $x:=x+1$) or even don't change it. Let's introduce the total pairwise distance — the sum of distances between each pair of friends. Let a' , b' and c' be the final positions of the first, the second and the third friend, correspondingly. Then the total pairwise distance is $|a'-b'|+|a'-c'|+|b'-c'|$, where $|x|$ is the absolute value of x .

Friends are interested in the minimum total pairwise distance they can reach if they will move optimally. Each friend will move no more than once. So, more formally, they want to know the minimum total pairwise distance they can reach after one minute.

You have to answer q independent test cases.

Input

The first line of the input contains one integer q ($1 \le q \le 1000$) the number of test cases. The next q lines describe test cases. The i-th test case is given as three integers a,b and c ($1 \le a,b,c \le 10$^9) initial positions of the first, second and third friend correspondingly. The positions of friends can be equal.

Output

For each test case print the answer on it — the minimum total pairwise distance (the minimum sum of distances between each pair of friends) if friends change their positions optimally. Each friend will move no more than once. So, more formally, you have to find the minimum total pairwise distance they can reach after one minute.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        for (int i=0;i<number;i++) {
            long one = input.nextLong();
            long two = input.nextLong();
            long three = input.nextLong();
            long average = (one + two + three) / 3;
            if (one == two && Math.abs(one - three)
== 2 || two == three && Math.abs(one - three) == 2 ||
one == three && Math.abs(one - two) == 2) {
                System.out.println("0");
            } else {
                if (one > average) {
                    one = one - 1;
                } else if (one < average) {
                    one = one + 1;
                }
                if (two > average) {
                    two = two - 1;
                } else if (two < average) {
                    two = two + 1;
                }
                if (three > average) {
                    three = three - 1;
                } else if (three < average) {
                    three = three + 1;
                }
                long distance = (Math.abs(one - two))
+ (Math.abs(one - three)) + (Math.abs(three - two));
                System.out.println(distance);
            }
        }
    }
}
```

The following problem is well-known: given integers *n* and *m*, calculate

$$2^n \bmod m,$$

where $2^n = 2 \cdot 2 \cdot \ldots \cdot 2$ (*n* factors), and $x \bmod y$ denotes the remainder of division of *x* by *y*.

You are asked to solve the "reverse" problem. Given integers *n* and *m*, calculate

$$m \bmod 2^n.$$

Input

The first line contains a single integer *n* ($1 \le n \le 10^8$).

The second line contains a single integer *m* ($1 \le m \le 10^8$).

Output

Output a single integer — the value of $m \bmod 2^n$.

Answer(java):

```
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int power= input.nextInt();
        int numbertoMUD= input.nextInt();

        double twotopower=Math.pow(2,power);

        double result=numbertoMUD%twotopower;

        System.out.println((int)result);
    }
}
```

A and B are preparing themselves for programming contests.

To train their logical thinking and solve problems better, A and B decided to play chess. During the game A wondered whose position is now stronger.

For each chess piece we know its weight:

- the queen's weight is 9,
- the rook's weight is 5,
- the bishop's weight is 3,
- the knight's weight is 3,
- the pawn's weight is 1,
- the king's weight isn't considered in evaluating position.

The player's weight equals to the sum of weights of all his pieces on the board.

As A doesn't like counting, he asked you to help him determine which player has the larger position weight.

Input

The input contains eight lines, eight characters each — the board's description.

The white pieces on the board are marked with uppercase letters, the black pieces are marked with lowercase letters.

The white pieces are denoted as follows: the queen is represented is 'Q', the rook — as 'R', the bishop — as'B', the knight — as 'N', the pawn — as 'P', the king — as 'K'.

The black pieces are denoted as 'q', 'r', 'b', 'n', 'p', 'k', respectively.

An empty square of the board is marked as '.' (a dot).

It is not guaranteed that the given chess position can be achieved in a real game. Specifically, there can be an arbitrary (possibly zero) number pieces of each type, the king may be under attack and so on.

Output

Print "White" (without quotes) if the weight of the position of the white pieces is more than the weight of the position of the black pieces, print "Black" if the weight of the black pieces is more than the weight of the white pieces and print "Draw" if the weights of the white and black pieces are equal.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Scanner;

public class Main {
        public static void main(String args[]) throws
IOException {
        int white=0;
        int black=0;
        Scanner input = new Scanner(System.in);
        for (int i=0;i<8;i++) {
            String word = input.nextLine();
            for (int j=0;j<8;j++){
                if (word.charAt(j)=='Q') {
                    white = white + 9;
                }else  if (word.charAt(j)=='R') {
                    white = white + 5;
                }else  if (word.charAt(j)=='B') {
                    white = white + 3;
                }else  if (word.charAt(j)=='N') {
                    white = white + 3;
                }else  if (word.charAt(j)=='P') {
                    white = white + 1;
                }
                if (word.charAt(j)=='q') {
                    black = black + 9;
                }else  if (word.charAt(j)=='r') {
                    black = black + 5;
                }else  if (word.charAt(j)=='b') {
                    black = black + 3;
                }else  if (word.charAt(j)=='n') {
                    black = black + 3;
                }else  if (word.charAt(j)=='p') {
                    black = black + 1;
                }
            }
        }
        if (black>white){
            System.out.println("Black");
        }else if (black==white){
            System.out.println("Draw");
        }else if (white>black){
            System.out.println("White");
        }
        }
}
```

Misha and Vasya participated in a Codeforces contest. Unfortunately, each of them solved only one problem, though successfully submitted it at the first attempt. Misha solved the problem that costs *a* points and Vasya solved the problem that costs *b* points. Besides, Misha submitted the problem *c* minutes after the contest started and Vasya submitted the problem *d* minutes after the contest started. As you know, on Codeforces the cost of a problem reduces as a round continues. That is, if you submit a problem that costs *p* points *t* minutes after the contest started, you get $\max\left(\frac{3p}{10}, p - \frac{p}{250} \times t\right)$points.

Misha and Vasya are having an argument trying to find out who got more points. Help them to find out the truth.

Input

The first line contains four integers *a, b, c, d* ($250 \le a, b \le 3500$, $0 \le c, d \le 180$).

It is guaranteed that numbers *a* and *b* are divisible by 250 (just like on any real Codeforces round).

Output

Output on a single line:

"Misha" (without the quotes), if Misha got more points than Vasya.

"Vasya" (without the quotes), if Vasya got more points than Misha.

"Tie" (without the quotes), if both of them got the same number of points.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int number1 = input.nextInt();
        int number2 = input.nextInt();
        int number3 = input.nextInt();

        int Misha=Math.max(3*number/10,number-
(number*number2)/250);
        int Vasya=Math.max(3*number1/10,number1-
(number1*number3)/250);


        if (Misha>Vasya){
            System.out.println("Misha");
        } else if (Vasya>Misha){
            System.out.println("Vasya");
        }else {
            System.out.println("Tie");
        }

    }
}
```

Today, Wet Shark is given *n* integers. Using any of these integers no more than once, Wet Shark wants to get maximum possible even (divisible by 2) sum. Please, calculate this value for Wet Shark.

Note, that if Wet Shark uses no integers from the *n* integers, the sum is an even integer 0.

Input

The first line of the input contains one integer, n $(1 \le n \le 100\,000)$. The next line contains *n* space separated integers given to Wet Shark. Each of these integers is in range from 1 to 10^9, inclusive.

Output

Print the maximum possible even sum that can be obtained if we use some of the given integers.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Vector;
public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        Long []numbers = new Long[number];
        long sum=0;
        for (int i=0;i<number;i++) {
            numbers[i] = input.nextLong();
            // even + even = even
            // odd + odd = even
            sum = sum + numbers[i];
        }
        if (sum%2==0){
            System.out.println(sum);
        }else {
            Arrays.sort(numbers);
            for (int j=0;j<number;j++){
                if (numbers[j]%2==1){
                    sum = sum - numbers[j];
                    break;
                }
            }
            System.out.println(sum);
        }
    }
}
```

Recently Vasya decided to improve his pistol shooting skills. Today his coach offered him the following exercise. He placed n cans in a row on a table. Cans are numbered from left to right from 1 to n. Vasya has to knock down each can exactly once to finish the exercise. He is allowed to choose the order in which he will knock the cans down. Vasya knows that the durability of the i-th can is a_i. It means that if Vasya has already knocked x cans down and is now about to start shooting the i-th one, he will need $(a_i \cdot x+1)$ shots to knock it down. You can assume that if Vasya starts shooting the i-th can, he will be shooting it until he knocks it down. Your task is to choose such an order of shooting so that the number of shots required to knock each of the n given cans down exactly once is minimum possible.

Input

The first line of the input contains one integer n $(2 \le n \le 1000)$ the number of cans. The second line of the input contains the sequence $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 1000)$, where a_i is the durability of the i-th can.

Output

In the first line print the minimum number of shots required to knock each of the n given cans down exactly once. In the second line print the sequence consisting of n distinct integers from 1 to n the order of indices of cans that minimizes the number of shots required. If there are several answers, you can print any of them.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        int[]numbers=new int[number];
        int[]numbers2=new int[number];
        int[]result=new int[number];
        int count =0;
        for (int i=0;i<number;i++){
            int nu= input.nextInt();
            numbers[i]=nu;
            numbers2[i]=nu;
        }
        Arrays.sort(numbers);
        for (int j=number-1;j>=0;j--){
            for (int k=0;k<number;k++){
                if (numbers[j]==numbers2[k]){
                    result[count]=k+1;
                    numbers2[k]=0;
                    count++;
                }
            }
        }
        int x=0;
        int result0=0;
        for (int z=0;z<number;z++){
         result0=result0+ (numbers[number-1-z]*x)+1;
            x++;
        }
        System.out.println(result0);
        for (int z2=0;z2<number;z2++){
            System.out.print(result[z2]+" ");
        }
    }
}
```

3. 1000 SCORE PROBLEMS

Pari has a friend who loves palindrome numbers. A palindrome number is a number that reads the same forward or backward. For example 12321, 100001 and 1 are palindrome numbers, while 112 and 1021 are not.

Pari is trying to love them too, but only very special and gifted people can understand the beauty behind palindrome numbers. Pari loves integers with even length (i.e. the numbers with even number of digits), so she tries to see a lot of big palindrome numbers with even length (like a 2-digit 11 or 6-digit 122221), so maybe she could see something in them.

Now Pari asks you to write a program that gets a huge integer n from the input and tells what is the n-th even-length positive palindrome number?

Input

The only line of the input contains a single integer n ($1 \leq n \leq 10^{100\,000}$).

Output

Print the n-th even-length palindrome number.

Answer(java):

```
package com.company;
import java.util.*;
import java.io.*;

public class Main {

    public static void main(String args[]) throws
IOException {
        Scanner read = new Scanner(System.in);
        String str = read.nextLine();
        String reverse = "";
        for(int i = str.length() - 1; i >= 0; i--)
        {
            reverse = reverse + str.charAt(i);
        }
        System.out.println(str+reverse);

    }
}
```

A bracket sequence is called regular if it is possible to obtain correct arithmetic expression by inserting characters + and 1 into this sequence. For example, sequences (())(), () and (()(())) are regular, while)(, (() and (()))(are not. Let's call a regular bracket sequence "RBS". You are given a sequence s of n characters (,), and/or ?. There is exactly one character (and exactly one character) in this sequence.

You have to replace every character ? with either) or ((different characters ? can be replaced with different brackets). You cannot reorder the characters, remove them, insert other characters, and each ? must be replaced.

Determine if it is possible to obtain an RBS after these replacements.

Input

The first line contains one integer t ($1 \le t \le 1000$) the number of test cases.

Each test case consists of one line containing s ($2 \le |s| \le 100$) a sequence of characters (,), and/or ?. There is exactly one character (and exactly one character) in this sequence.

Output

For each test case, print YES if it is possible to obtain a regular bracket sequence, or NO otherwise}. You may print each letter in any case (for example, YES, Yes, yes, yEs will all be recognized as positive answer).

Answer(java):

```java
package com.company;
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        String num = input.nextLine();
        int number = Integer.parseInt(num);
        int open = 0;
        int close = 0;
        int quastion = 0;
        boolean openAtFirst = false;
        for (int i = 0; i < number; i++) {
            String word = input.nextLine();
            openAtFirst=false;
            if (word.length() % 2 == 1) {
                System.out.println("NO");
            } else {
                for (int j = 0; j < word.length();
j++) {
                    if (word.charAt(0) == ')' ||
word.charAt(word.length() - 1) == '(') {
                        System.out.println("NO");
                        openAtFirst = true;
                        break;
                    } else if (word.charAt(j) == '(')
{
                        open = open + 1;
                    } else if (word.charAt(j) == ')')
{
                        close = close + 1;
                    } else if (word.charAt(j) == '?')
{
                        quastion = quastion + 1;
                    }
                }
                if (!openAtFirst) {
                    if (Math.abs(open - close) ==
quastion || open - close == 0) {
                        System.out.println("YES");
                        open = 0;
                        close = 0;
                        quastion = 0;
                    } else {
                        System.out.println("NO");
                        open = 0;
                        close = 0;
                        quastion = 0;
                    }
                }
            }
        }
    }
}
```

There are two rival donut shops. The first shop sells donuts at retail: each donut costs a dollars. The second shop sells donuts only in bulk: box of b donuts costs c dollars. So if you want to buy x donuts from this shop, then you have to buy the smallest number of boxes such that the total number of donuts in them is greater or equal to x.

You want to determine two positive integer values:

1. how many donuts can you buy so that they are strictly cheaper in the first shop than in the second shop?
2. how many donuts can you buy so that they are strictly cheaper in the second shop than in the first shop?

If any of these values doesn't exist then that value should be equal to -1. If there are multiple possible answers, then print any of them. The printed values should be less or equal to 10^9. It can be shown that under the given constraints such values always exist if any values exist at all.

Input

The first line contains a single integer t ($1 \le t \le 1000$) the number of testcases.

Each of the next t lines contains three integers a, b and c ($1 \le a \le 10$^9, $2 \le b \le 10$^9, $1 \le c \le 10$^9).

Output

For each testcase print two positive integers. For both shops print such x that buying x donuts in this shop is strictly cheaper than buying x donuts in the other shop. x should be greater than 0 and less or equal to 10^9. If there is no such x, then print -1. If there are multiple answers, then print any of them.

Answer(python):

```
for i in range (int(input())):
        a, b, c = map(int, input().split())

        print(1 if a<c else -1,end=" ")
    print(b if c<a*b else -1)
```

Mahmoud has n line segments, the i-th of them has length a_i. Ehab challenged him to use exactly 3 line segments to form a non-degenerate triangle. Mahmoud doesn't accept challenges unless he is sure he can win, so he asked you to tell him if he should accept the challenge. Given the lengths of the line segments, check if he can choose exactly 3 of them to form a non-degenerate triangle.

Mahmoud should use exactly 3 line segments, he can't concatenate two line segments or change any length. A non-degenerate triangle is a triangle with positive area.

Input

The first line contains single integer n ($3 \le n \le 10^5$) — the number of line segments Mahmoud has.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$) — the lengths of line segments Mahmoud has.

Output

In the only line print "YES" if he can choose exactly three line segments and form a non-degenerate triangle with them, and "NO" otherwise.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input=new Scanner(System.in);
        int number = input.nextInt();
        int [] arr=new int[number];
        for (int i=0;i<number;i++){
            arr[i] = input.nextInt();
        }
        Arrays.sort(arr);
        boolean check=false;
        //System.out.println(Arrays.toString(arr));
        for (int j=0;j<number-2;j++){

            if (arr[j]+arr[j+1]>arr[j+2]){
                System.out.println("YES");
                check=true;
                break;
            }
        }
        if (!check){
            System.out.println("NO");
        }
    }
}
```

Recently a dog was bought for Polycarp. The dog's name is Cormen. Now Polycarp has a lot of troubles. For example, Cormen likes going for a walk.

Empirically Polycarp learned that the dog needs at least k walks for any two consecutive days in order to feel good. For example, if $k = 5$ and yesterday Polycarp went for a walk with Cormen 2 times, today he has to go for a walk at least 3 times.

Polycarp analysed all his affairs over the next n days and made a sequence of n integers $a_1, a_2, ..., a_n$, where a_i is the number of times Polycarp will walk with the dog on the i-th day while doing all his affairs (for example, he has to go to a shop, throw out the trash, etc.).

Help Polycarp determine the minimum number of walks he needs to do additionaly in the next n days so that Cormen will feel good during all the n days. You can assume that on the day before the first day and on the day after the n-th day Polycarp will go for a walk with Cormen exactly k times.

Write a program that will find the minumum number of additional walks and the appropriate schedule — the sequence of integers $b_1, b_2, ..., b_n$ ($b_i \geq a_i$), where b_i means the total number of walks with the dog on the i-th day.

Input

The first line contains two integers n and k ($1 \leq n, k \leq 500$) — the number of days and the minimum number of walks with Cormen for any two consecutive days.

The second line contains integers $a_1, a_2, ..., a_n$ ($0 \leq a_i \leq 500$) — the number of walks with Cormen on the i-th day which Polycarp has already planned.

Output

In the first line print the smallest number of additional walks that Polycarp should do during the next n days so that Cormen will feel good during all days.

In the second line print n integers $b_1, b_2, ..., b_n$, where b_i — the total number of walks on the i-th day according to the found solutions ($a_i \leq b_i$ for all i from 1 to n). If there are multiple solutions, print any of them.

Answer(java):

```
package com.company;
import java.util.*;
import java.io.*;

public class Main {

    public static void main(String args[]) throws
IOException {
        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int k = input.nextInt();
        int [] arr = new int[n];
        int total=0;
        for (int i=0;i<n;i++){
            arr[i]= input.nextInt();
        }

        for (int j=0;j<n;j++){
            if (j+1==n){
                continue;
            }
            if (arr[j]+arr[j+1]<k){
                int difference= k-(arr[j]+arr[j+1]);
                arr[j+1]=arr[j+1]+difference;
                total = total +difference;
            }
        }
        System.out.println(total);
        for (int z=0;z<n;z++){
            System.out.print(arr[z]+"  ");
        }
    }
}
```

Vasya's birthday is approaching and Lena decided to sew a patterned handkerchief to him as a present. Lena chose digits from 0 to *n* as the pattern. The digits will form a rhombus. The largest digit *n* should be located in the centre. The digits should decrease as they approach the edges. For example, for $n = 5$ the handkerchief pattern should look like that:

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

Your task is to determine the way the handkerchief will look like by the given *n*.

Input

The first line contains the single integer *n* ($2 \le n \le 9$).

Output

Print a picture for the given *n*. You should strictly observe the number of spaces before the first digit on each line. Every two adjacent digits in the same line should be separated by exactly one space. There should be no spaces after the last digit at the end of each line.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        switch (number) {
            case 1:
                System.out.println(
                        "  0\n" +
                                "0 1 0\n" +
                                "  0\n");
                break;
            case 2:
                System.out.println(
                        "    0\n" +
                                "  0 1 0\n" +
                                "0 1 2 1 0\n" +
                                "  0 1 0\n" +
                                "    0\n");
                break;

            case 3:
                System.out.println(
                        "      0\n" +
                                "    0 1 0\n" +
                                "  0 1 2 1 0\n" +
                                "0 1 2 3 2 1 0\n" +
                                "  0 1 2 1 0\n" +
                                "    0 1 0\n" +
                                "      0\n");
                break;

            case 4:
                System.out.println(
                        "        0\n" +
                                "      0 1 0\n" +
                                "    0 1 2 1 0\n" +
                                "  0 1 2 3 2 1 0\n" +
                                "0 1 2 3 4 3 2 1 0\n"
+
                                "  0 1 2 3 2 1 0\n" +
                                "    0 1 2 1 0\n" +
                                "      0 1 0\n" +
                                "        0\n");

                break;
            case 5:
                System.out.println(
                        "          0\n" +
                                "        0 1 0\n" +
                                "      0 1 2 1 0\n" +
                                "    0 1 2 3 2 1 0\n"
+
                                "  0 1 2 3 4 3 2 1
0\n" +
                                "0 1 2 3 4 5 4 3 2 1
0\n" +
                                "  0 1 2 3 4 3 2 1
0\n" +
                                "    0 1 2 3 2 1 0\n"
+
                                "      0 1 2 1 0\n" +
                                "        0 1 0\n" +
                                "          0\n");
                break;
            case 6:
                System.out.println(
                        "            0\n" +
                                "          0 1 0\n" +
                                "        0 1 2 1 0\n"
+
                                "      0 1 2 3 2 1
0\n" +
                                "    0 1 2 3 4 3 2 1
0\n" +
                                "  0 1 2 3 4 5 4 3 2
1 0\n" +
                                "0 1 2 3 4 5 6 5 4 3
2 1 0\n" +
                                "  0 1 2 3 4 5 4 3 2
1 0\n" +
                                "    0 1 2 3 4 3 2 1
0\n" +
                                "      0 1 2 3 2 1
0\n" +
                                "        0 1 2 1 0\n"
+
                                "          0 1 0\n" +
                                "            0\n");
                break;

            case 7:
                System.out.println(
                        "              0\n" +
                                "            0 1 0\n"
+
                                "          0 1 2 1
0\n" +
                                "        0 1 2 3 2 1
0\n" +
                                "      0 1 2 3 4 3 2
1 0\n" +
                                "    0 1 2 3 4 5 4 3
2 1 0\n" +
                                "  0 1 2 3 4 5 6 5 4
3 2 1 0\n" +
                                "0 1 2 3 4 5 6 7 6 5
4 3 2 1 0\n" +
                                "  0 1 2 3 4 5 6 5 4
3 2 1 0\n" +
                                "    0 1 2 3 4 5 4 3
2 1 0\n" +
                                "      0 1 2 3 4 3 2
1 0\n" +
                                "        0 1 2 3 2 1
0\n" +
                                "          0 1 2 1
0\n" +
                                "            0 1 0\n"
+
                                "              0\n");
                break;

            case 8:
                System.out.println(
                        "                0\n" +
                                "              0 1
0\n" +
                                "            0 1 2 1
0\n" +
                                "          0 1 2 3 2
1 0\n" +
                                "        0 1 2 3 4 3
2 1 0\n" +
                                "      0 1 2 3 4 5 4
3 2 1 0\n" +
                                "    0 1 2 3 4 5 6 5
4 3 2 1 0\n" +
                                "  0 1 2 3 4 5 6 7 6
5 4 3 2 1 0\n" +
                                "0 1 2 3 4 5 6 7 8 7
6 5 4 3 2 1 0\n" +
                                "  0 1 2 3 4 5 6 7 6
5 4 3 2 1 0\n" +
                                "    0 1 2 3 4 5 6 5
4 3 2 1 0\n" +
                                "      0 1 2 3 4 5 4
3 2 1 0\n" +
                                "        0 1 2 3 4 3
2 1 0\n" +
                                "          0 1 2 3 2
1 0\n" +
                                "            0 1 2 1
0\n" +
                                "              0 1
0\n" +
                                "
0\n");
                break;

            case 9:
                System.out.println(
                        "                  0\n" +
                                "                0 1
0\n" +
                                "              0 1 2
1 0\n" +
                                "            0 1 2 3
2 1 0\n" +
                                "          0 1 2 3 4
3 2 1 0\n" +
                                "        0 1 2 3 4 5
4 3 2 1 0\n" +
                                "      0 1 2 3 4 5 6
5 4 3 2 1 0\n" +
                                "    0 1 2 3 4 5 6 7
6 5 4 3 2 1 0\n" +
                                "  0 1 2 3 4 5 6 7 8
7 6 5 4 3 2 1 0\n" +
                                "0 1 2 3 4 5 6 7 8 9
8 7 6 5 4 3 2 1 0\n" +
                                "  0 1 2 3 4 5 6 7 8
7 6 5 4 3 2 1 0\n" +
                                "    0 1 2 3 4 5 6 7
6 5 4 3 2 1 0\n" +
                                "      0 1 2 3 4 5 6
5 4 3 2 1 0\n" +
                                "        0 1 2 3 4 5
4 3 2 1 0\n" +
                                "          0 1 2 3 4
3 2 1 0\n" +
                                "            0 1 2 3
2 1 0\n" +
                                "              0 1 2
1 0\n" +
                                "                0 1
0\n" +
                                "
0\n");
                break;
        }
    }
}
```

Polycarp plays "Game 23". Initially he has a number n and his goal is to transform it to m. In one move, he can multiply n by 2 or multiply n by 3. He can perform any number of moves.

Print the number of moves needed to transform n to m. Print -1 if it is impossible to do so. It is easy to prove that any way to transform n to m contains the same number of moves (i.e. number of moves doesn't depend on the way of transformation).

Input

The only line of the input contains two integers n and m ($1 \le n \le m \le 5 \cdot 10$^8).

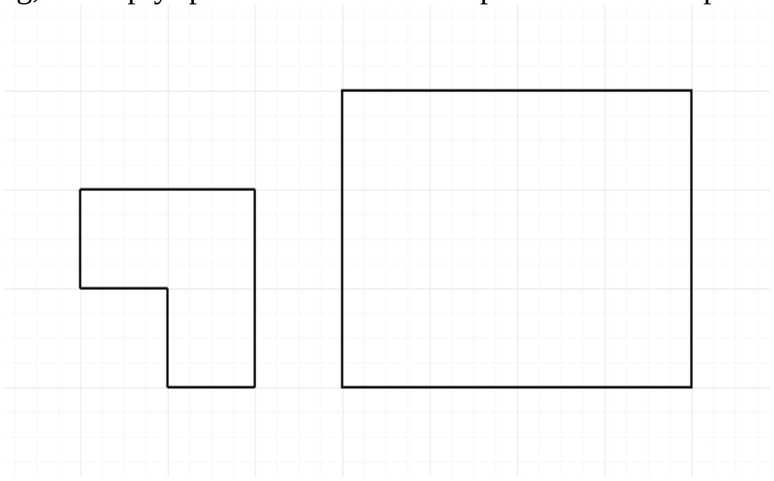
Output

Print the number of moves to transform n to m, or -1 if there is no solution.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input=new Scanner(System.in);
        int number1= input.nextInt();
        int number2= input.nextInt();
        int d=number2/number1;
        int sum2=0;
        int sum3=0;

        if (number2%number1==0){
            while (d%2==0){
                d=d/2;
                sum2=sum2+1;
            }
            while (d%3==0){
                d=d/3;
                sum3=sum3+1;
            }
            if (d!=1){
                System.out.println("-1");
            }else {
                System.out.println(sum2 + sum3);
            }

        }else {
            System.out.println("-1");
        }
    }
}
```

You have a given integer n. Find the number of ways to fill all $3 \times n$ tiles with the shape described in the picture below. Upon filling, no empty spaces are allowed. Shapes cannot overlap.



This picture describes when $n=4$.
The left one is the shape and the right one is $3 \times n$ tiles.

Input

The only line contains one integer n ($1 \le n \le 60$) the length.

Output

Print the number of ways to fill.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        if (number%2==1){
            System.out.println("0");
        }else {

System.out.println((int)Math.pow(2,number/2));
        }
    }
}
```

Valera runs a 24/7 fast food cafe. He magically learned that next day n people will visit his cafe. For each person we know the arrival time: the i-th person comes exactly at h_i hours m_i minutes. The cafe spends less than a minute to serve each client, but if a client comes in and sees that there is no free cash, than he doesn't want to wait and leaves the cafe immediately.

Valera is very greedy, so he wants to serve all n customers next day (and get more profit). However, for that he needs to ensure that at each moment of time the number of working cashes is no less than the number of clients in the cafe.

Help Valera count the minimum number of cashes to work at his cafe next day, so that they can serve all visitors.

Input

The first line contains a single integer n ($1 \le n \le 10^5$), that is the number of cafe visitors.

Each of the following n lines has two space-separated integers h_i and m_i ($0 \le h_i \le 23$; $0 \le m_i \le 59$), representing the time when the i-th person comes into the cafe.

Note that the time is given in the chronological order. All time is given within one 24-hour period.

Output

Print a single integer the minimum number of cashes, needed to serve all clients next day.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int[] hours = new int[number];
        int[] min = new int[number];
        int[] sums = new int[number+1];
        int[] last = new int[number + 1];
        int sum = 1;
        int count = 0;
        for (int i = 0; i < number; i++) {
            hours[i] = input.nextInt();
            min[i] = input.nextInt();
        }
        for (int j = 0; j < number; j++) {
            if (j+1==number){
                continue;
            }
            if (hours[j]==hours[j+1] &&
min[j]==min[j+1]){
                sum=sum+1;

            }else {
                sums[j]=sum;
                sum=1;
            }
        }
        sums[number]=sum;
        Arrays.sort(sums);
        //System.out.println(Arrays.toString(sums));
        if (sums[number] == 0) {
            System.out.println("1");
        } else {
            System.out.println(sums[number]);
        }
    }
}
```

Little Vasya has received a young builder's kit. The kit consists of several wooden bars, the lengths of all of them are known. The bars can be put one on the top of the other if their lengths are the same.

Vasya wants to construct the minimal number of towers from the bars. Help Vasya to use the bars in the best way possible.

Input

The first line contains an integer N ($1 \leq N \leq 1000$) — the number of bars at Vasya's disposal. The second line contains N space-separated integers l_i — the lengths of the bars. All the lengths are natural numbers not exceeding 1000.

Output

In one line output two numbers — the height of the largest tower and their total number. Remember that Vasya should use all the bars.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner z=new Scanner(System.in);
        int a=z.nextInt();
        int[] b=new int[a];
        for(int i=0;i<b.length;i++)
            b[i]=z.nextInt();
        int k1=0,y=0;
        for(int i=0;i<b.length;i++)
        {
            int k=0;
            for(int j=i;j<b.length;j++)
            {
                if(b[i]==b[j])
                    k++;
            }
            if(k==1)
                k1++;
            y=Math.max(y, k);
        }
        System.out.println(y+" "+k1);
    }
}
```

Jeff's got *n* cards, each card contains either digit 0, or digit 5. Jeff can choose several cards and put them in a line so that he gets some number. What is the largest possible number divisible by 90 Jeff can make from the cards he's got?

Jeff must make the number without leading zero. At that, we assume that number 0 doesn't contain any leading zeroes. Jeff doesn't have to use all the cards.

Input

The first line contains integer n ($1 \leq n \leq 10^3$). The next line contains n integers a_1, a_2, ..., a_n ($a_i = 0$ or $a_i = 5$). Number a_i represents the digit that is written on the i-th card.

Output

In a single line print the answer to the problem — the maximum number, divisible by 90. If you can't make any divisible by 90 number from the cards, print -1.

Answer(java):

```
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class main {

    public static void main(String[]args){
        Scanner input = new Scanner(System.in);

        int n = input.nextInt();
        int t;
        int z=0;
        int f=0;
        for (int i=0;i<n;i++){
            t= input.nextInt();
            if (t==0){
                z++;
            }else if (t==5){
                f++;
            }
        }

        if (z==0){
            System.out.print("-1");
        }else if (f<9){
            System.out.print("0");
        }else {
            f-=f%9;
            for (int j=0;j<f;j++){
                System.out.print("5");
            }
            for (int k=0;k<z;k++){
                System.out.print("0");
            }
        }

    }
}
```

4 1100 SCORE PROBLEMS

The new "Die Hard" movie has just been released! There are n people at the cinema box office standing in a huge line. Each of them has a single 100, 50 or 25 ruble bill. A "Die Hard" ticket costs 25 rubles. Can the booking clerk sell a ticket to each person and give the change if he initially has no money and sells the tickets strictly in the order people follow in the line?

Input

The first line contains integer n ($1 \le n \le 10^5$) — the number of people in the line. The next line contains n integers, each of them equals 25, 50 or 100 — the values of the bills the people have. The numbers are given in the order from the beginning of the line (at the box office) to the end of the line.

Output

Print "YES" (without the quotes) if the booking clerk can sell a ticket to each person and give the change. Otherwise print "NO".

Answer 1(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int sum25=0;
        int sum50=0;
        int no=0;
        for (int i = 0; i < number; i++) {
            int money = input.nextInt();
            if (number==80001){
                no=no-1;
            }
            if (number==80003){
                no=no-1;
            }
            if (money==25){
                sum25=sum25+1;
            }else if (money==50){
                sum50=sum50+1;
            }
            if (money==50 && sum25>0){
                sum25=sum25-1;
         }else if (money==100 && sum50>0 && sum25>0){
                sum50=sum50-1;
                sum25=sum25-1;
            }else if (money==100 && sum25>1){
                sum25=sum25-2;
            }else if (money==50 && sum25==0){
                no=no+1;
            }else if (money==100 && sum50==0){
                no=no+1;
            }else if (money==100 && sum25==0){
                no=no+1;
            }else if(money==100&&sum25<1&&sum50<1){
                no=no+1;
            }
        }
        if (no==0){
            System.out.println("YES");
        }else {
            System.out.println("NO");
            }
        }
}
```

Answer 2(java):

```
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int sum25=0;
        int sum50=0;
        int no=0;
        for (int i = 0; i < number; i++) {
            int money = input.nextInt();
            if (number==80001){
                no=no-1;
            }
            if (number==80003){
                no=no-1;
            }
            if (money==25){
                sum25=sum25+1;
            }else if (money==50){
                sum50=sum50+1;
            }
            if (money==50 && sum25>0){
                sum25=sum25-1;
         }else if (money==100 && sum50>0 && sum25>0){
                sum50=sum50-1;
                sum25=sum25-1;
            }else if (money==100 && sum25>1){
                sum25=sum25-2;
            }else if (money==50 && sum25==0){
                no=no+1;
            }else if (money==100 && sum50==0){
                no=no+1;
            }else if (money==100 && sum25==0){
                no=no+1;
            }else if(money==100&&sum25<1&&sum50<1){
                no=no+1;
            }
        }
        if (no==0){
            System.out.println("YES");
        }else {
            System.out.println("NO");
        }
    }
}
```

We consider a positive integer perfect, if and only if the sum of its digits is exactly 10. Given a positive integer k, your task is to find the k-th smallest perfect positive integer.

Input

A single line with a positive integer $k (1 \le k \le 10000)$.

Output

A single number, denoting the k-th smallest perfect integer.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Scanner;

public class Main {
    /* Function to get sum of digits */
    static long getSum(long n)
    {
        long sum;

        /* Single line that calculates sum */
        for (sum = 0; n > 0; sum += n % 10,
                n /= 10);

        return sum;
    }
    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number= input.nextInt();
        int sum=0;
        int count=0;
        for (int n=10;n<1000000000;n++) {

            if (getSum(n)==10){
                count=count+1;
            }
            if (count==number){
                System.out.println(n);
                break;
            }
        }
    }
}
```

Alice has a cute cat. To keep her cat fit, Alice wants to design an exercising walk for her cat!

Initially, Alice's cat is located in a cell (x,y)

of an infinite grid. According to Alice's theory, cat needs to move:

- exactly a

steps left: from (u,v) to $(u-1,v)$
- ;
- exactly b steps right: from (u,v) to $(u+1,v)$
- ;
- exactly c steps down: from (u,v) to $(u,v-1)$
- ;
- exactly d steps up: from (u,v) to $(u,v+1)$

- .

Note that the moves can be performed in an arbitrary order. For example, if the cat has to move 1

step left, 3 steps right and 2

steps down, then the walk right, down, left, right, right, down is valid.

Alice, however, is worrying that her cat might get lost if it moves far away from her. So she hopes that her cat is always in the area $[x_1,x_2]\times[y_1,y_2]$

, i.e. for every cat's position (u,v) of a walk $x_1\le u\le x_2$ and $y_1\le v\le y_2$

holds.

Also, note that the cat can visit the same cell multiple times.

Can you help Alice find out if there exists a walk satisfying her wishes?

Formally, the walk should contain exactly $a+b+c+d$ unit moves (a to the left, b to the right, c to the down, d to the up). Alice can do the moves in any order. Her current position (u,v) should always satisfy the constraints: $x_1 \le u \le x_2$, $y_1 \le v \le y_2$. The staring point is (x,y).

You are required to answer t test cases independently.

Input

The first line contains a single integer t ($1 \le t \le 10$^3) the number of testcases. The first line of each test case contains four integers a, b, c, d ($0 \le a,b,c,d \le 10_8$, $a+b+c+d \ge 1$). The second line of the test case contains six integers x, y, x_1, y_1, x_2, y_2 ($-10^8\le x_1 \le x \le x_2 \le 10$^8, $-10^8\le y_1 \le y \le y_2 \le 10$^8).

Output

For each test case, output "YES" in a separate line, if there exists a walk satisfying her wishes. Otherwise, output "NO" in a separate line.

You can print each letter in any case (upper or lower).

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        for (int i=0;i<number;i++){
            int a = input.nextInt();
            int b = input.nextInt();
            int c = input.nextInt();
            int d = input.nextInt();
            int x = input.nextInt();
            int y = input.nextInt();
            int x1 = input.nextInt();
            int y1 = input.nextInt();
            int x2 = input.nextInt();
            int y2 = input.nextInt();
            x+=b-a;
            y+=d-c;
            if (a+b>0 && x1 == x2){
                System.out.println("NO");
            }else if (c+d>0 && y1==y2){
                System.out.println("NO");
        }else if (x1<= x && x<=x2 && y1<=y && y<=y2){
                System.out.println("YES");
            }else {
                System.out.println("NO");
            }
        }
    }
}
```

You are playing a new computer game in which you have to fight monsters. In a dungeon you are trying to clear, you met three monsters; the first of them has a health points, the second has b health points, and the third has c.

To kill the monsters, you can use a cannon that, when fired, deals 1damage to the selected monster. Every 7-th (i. e. shots with numbers 7, 14, 21 etc.) cannon shot is enhanced and deals 1 damage to all monsters, not just one of them. If some monster's current amount of health points is 0, it can't be targeted by a regular shot and does not receive damage from an enhanced shot.

You want to pass the dungeon beautifully, i. e., kill all the monsters with the same enhanced shot (i. e. after some enhanced shot, the health points of each of the monsters should become equal to 0 for the first time). Each shot must hit a monster, i. e. each shot deals damage to at least one monster.

Input

The first line contains a single integer t ($1 \le t \le 10$^4) the number of test cases.Each test case consists of a single line that contains three integers a, b and c ($1 \le a,b,c \le 10$^8) the number of health points each monster has.

Output

For each test case, print YES if you can kill all the monsters with the same enhanced shot. Otherwise, print NO. You may print each letter in any case (for example, YES, Yes, yes, yEs will all be recognized as positive answer).

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int sum=0;
        for (int i=0;i<number;i++){
            int dragon1 = input.nextInt();
            int dragon2 = input.nextInt();
            int dragon3 = input.nextInt();
            //System.out.println("result:
"+(dragon1+dragon2+dragon3)/9);
            if ((dragon1+dragon2+dragon3)/9<=dragon1
&& (dragon1+dragon2+dragon3)/9<=dragon2 &&
(dragon1+dragon2+dragon3)/9<=dragon3 &&
(dragon1+dragon2+dragon3)%9==0){
                System.out.println("YES");
            }else {
                System.out.println("NO");
            }
        }
    }
}
```

Caisa solved the problem with the sugar and now he is on the way back to home.

Caisa is playing a mobile game during his path. There are $(n + 1)$ pylons numbered from 0 to n in this game. The pylon with number 0 has zero height, the pylon with number i $(i > 0)$ has height h_i. The goal of the game is to reach n-th pylon, and the only move the player can do is to jump from the current pylon (let's denote its number as k) to the next one (its number will be $k + 1$). When the player have made such a move, its energy increases by $h_k - h_{k+1}$ (if this value is negative the player loses energy). The player must have non-negative amount of energy at any moment of the time.

Initially Caisa stand at 0 pylon and has 0 energy. The game provides a special opportunity: one can pay a single dollar and increase the height of anyone pylon by one. Caisa may use that opportunity several times, but he doesn't want to spend too much money. What is the minimal amount of money he must paid to reach the goal of the game?

Input

The first line contains integer n $(1 \le n \le 10^5)$. The next line contains n integers $h_1, h_2, ..., h_n$ $(1 \le h_i \le 10^5)$ representing the heights of the pylons.

Output

Print a single number representing the minimum number of dollars paid by Caisa.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int n = input.nextInt();
        int[] numbers=new int[n];

        for (int i = 0; i < n; i++) {
            numbers[i]=input.nextInt();
        }
        Arrays.sort(numbers);
        System.out.println(numbers[n-1]);
    }
}
```

Alexandra has an even-length array a, consisting of 0s and 1s. The elements of the array are enumerated from 1 to n. She wants to remove at most $n2$ elements (where n — length of array) in the way that alternating sum of the array will be equal 0 (i.e. $a_1-a_2+a_3-a_4+\ldots=0$). In other words, Alexandra wants sum of all elements at the odd positions and sum of all elements at the even positions to become equal. The elements that you remove don't have to be consecutive. For example, if she has $a=[1,0,1,0,0,0]$ and she removes 2nd and 4th elements, a will become equal $[1,1,0,0]$ and its alternating sum is $1-1+0-0=0$. Help her!

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1\le t\le 10$^3). Description of the test cases follows.

The first line of each test case contains a single integer n ($2\le n\le 10$3, n is even) length of the array. The second line contains n integers $a_1,a_2,\ldots,a_n$ ($0\le a_i\le 1$) elements of the array. It is guaranteed that the sum of n over all test cases does not exceed 10^3.

Output

For each test case, firstly, print k ($n2\le k\le n$) — number of elements that will remain after removing in the order they appear in a. Then, print this k numbers. Note that you should print the numbers themselves, not their indices. We can show that an answer always exists. If there are several answers, you can output any of them.

Answer(java):

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int num = input.nextInt();
        int sum = 0;
        int sum0 = 0;
        for (int i = 0; i < num; i++) {
            int number = input.nextInt();
            for (int j = 0; j < number; j++) {
                int numbers = input.nextInt();
                if (numbers == 1) {
                    sum = sum + 1;
                } else {
                    sum0 = sum0 + 1;
                }
            }
            if (sum <= number / 2) {
                System.out.println(sum0);
                for (int k = 0; k < sum0; k++) {
                    System.out.print("0 ");
                }
                System.out.println(" ");
                sum = 0;
                sum0 = 0;
            } else {
                if (sum % 2 == 0) {
                    System.out.println(sum);
                    for (int k = 0; k < sum; k++) {
                        System.out.print("1 ");
                    }
                    System.out.println(" ");
                } else {
                    System.out.println(sum - 1);
                    for (int k = 0; k < sum -1; k++)
{
                        System.out.print("1 ");
                    }
                    System.out.println(" ");
                }
                sum = 0;
                sum0 = 0;
            }
        }
        System.out.println(" ");
    }
}
```

You are given three integers a, b and c. Find two positive integers x and y ($x>0, y>0$) such that:

- the decimal representation of x without leading zeroes consists of a digits;
- the decimal representation of y without leading zeroes consists of b digits;
- the decimal representation of $gcd(x,y)$ without leading zeroes consists of c digits.

$gcd(x,y)$ denotes the **greatest common divisor (GCD)** of integers x and y. Output x and y. If there are multiple answers, output any of them.

Input

The first line contains a single integer $t(1\le t\le 285)$ the number of testcases. Each of the next t lines contains three integers a, b and c ($1\le a,b\le 9$, $1\le c\le min(a,b)$) the required lengths of the numbers. It can be shown that the answer exists for all testcases under the given constraints. Additional constraint on the input: all testcases are different.

Output

For each testcase print two positive integers x and y ($x>0$, $y>0$) such that

- the decimal representation of x without leading zeroes consists of a digits;

- the decimal representation of y without leading zeroes consists of b digits;
- the decimal representation of $gcd(x,y)$ without leading zeroes consists of c digits.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        for (int i=0;i<number;i++){
            int a= input.nextInt();
            int b= input.nextInt();
            int c= input.nextInt();

            System.out.print("1");
            for (int z=0;z<a-1;z++){
                System.out.print("0");
            }
            System.out.print(" ");
            for (int k=0;k<b-c+1;k++){
                System.out.print("1");
            }
            for (int k1=0;k1<c-1;k1++){
                System.out.print("0");
            }
            System.out.println("");
        }
    }
}
```

Alice and Bob play ping-pong with simplified rules. During the game, the player serving the ball commences a play. The server strikes the ball then the receiver makes a return by hitting the ball back. Thereafter, the server and receiver must alternately make a return until one of them doesn't make a return. The one who doesn't make a return loses this play. The winner of the play commences the next play. Alice starts the first play. Alice has x stamina and Bob has y. To hit the ball (while serving or returning) each player spends 1 stamina, so if they don't have any stamina, they can't return the ball (and lose the play) or can't serve the ball (in this case, the other player serves the ball instead). If both players run out of stamina, the game is over. Sometimes, it's strategically optimal not to return the ball, lose the current play, but save the stamina. On the contrary, when the server commences a play, they have to hit the ball, if they have some stamina left. Both Alice and Bob play optimally and want to, firstly, maximize their number of wins and, secondly, minimize the number of wins of their opponent. Calculate the resulting number of Alice's and Bob's wins.

Input

The first line contains a single integer t ($1 \le t \le 10$^4) the number of test cases. The first and only line of each test case contains two integers x and y ($1 \le x,y \le 10$^6) Alice's and Bob's initial stamina.

Output

For each test case, print two integers — the resulting number of Alice's and Bob's wins, if both of them play optimally.

```
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        for (int i=0;i<number;i++){
            int x= input.nextInt();
            System.out.println(x-1);
            int y= input.nextInt();
            System.out.println(y);
        }
    }
}
```

The Little Elephant has an integer *a*, written in the binary notation. He wants to write this number on a piece of paper.

To make sure that the number *a* fits on the piece of paper, the Little Elephant ought to delete exactly one any digit from number *a* in the binary record. At that a new number appears. It consists of the remaining binary digits, written in the corresponding order (possible, with leading zeroes).

The Little Elephant wants the number he is going to write on the paper to be as large as possible. Help him find the maximum number that he can obtain after deleting exactly one binary digit and print it in the binary notation.

Input

The single line contains integer *a*, written in the binary notation without leading zeroes. This number contains more than 1 and at most 10^5 digits.

Output

In the single line print the number that is written without leading zeroes in the binary notation — the answer to the problem.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Vector;



public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        String a1 =input.nextLine();
        StringBuilder a= new StringBuilder(a1);

        for (int i=0;i<a.length();i++){
            if(a.charAt(i)=='0'){
                a = a.deleteCharAt(i);
                break;
            }else if (a.charAt(a.length()-1)=='1' &&
i==a.length()-1){
                a = a.deleteCharAt(i);
            }
        }
        System.out.println(a);
    }
}
```

Initially, you have the array a consisting of one element 1 ($a=[1]$).

In one move, you can do one of the following things:

- Increase some (single) element of a by 1 (choose some i from 1 to the current length of a and increase a_i by one);
- Append the copy of some (single) element of a to the end of the array (choose some i from 1 to the current length of a and append a_i to the end of the array).

For example, consider the sequence of five moves:

1. You take the first element a_1, append its copy to the end of the array and get $a=[1,1]$.
2. You take the first element a_1, increase it by 1 and get $a=[2,1]$.
3. You take the second element a_2, append its copy to the end of the array and get $a=[2,1,1]$.
4. You take the first element a_1, append its copy to the end of the array and get $a=[2,1,1,2]$.
5. You take the fourth element a_4, increase it by 1 and get $a=[2,1,1,3]$.

Your task is to find the minimum number of moves required to obtain the array with the sum at least n. You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \leq t \leq 1000$) the number of test cases. Then t test cases follow. The only line of the test case contains one integer n ($1 \leq n \leq 10$^9) the lower bound on the sum of the array.

Output

For each test case, print the answer: the minimum number of moves required to obtain the array with the sum at least n.

Answer(java):

```
package com.company;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Vector;
public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);

        int t = input.nextInt();
        while (t>0){
            int n = input.nextInt();
            int ans=1000000000;
            for (int x=1;x*x<=n;x++){
ans = Math.min(ans,x-1+((n-x)+x-1)/x);
            }
            t--;
            System.out.println(ans);
        }
    }
}
```

5 1200 SCORE PROBLEMS

You have array of n numbers $a_1,a_2,\ldots,a_n$. Rearrange these numbers to satisfy $|a_1-a_2|\le|a_2-a_3|\le\ldots\le|a_{n-1}-a_n|$, where $|x|$ denotes absolute value of x. It's always possible to find such rearrangement. Note that all numbers in a are not necessarily different. In other words, some numbers of a may be same. You have to answer independent t test cases.

Input

The first line contains a single integer t ($1\le t\le 10$^4) — the number of test cases. The first line of each test case contains single integer n ($3\le n\le 10$5) — the length of array a. It is guaranteed that the sum of values of n over all test cases in the input does not exceed 10^5. The second line of each test case contains n integers $a_1,a_2,\ldots,a_n$ ($-10^9\le a_i\le 10$^9).

Output

For each test case, print the rearranged version of array a which satisfies given condition. If there are multiple valid rearrangements, print any of them.

Answer(java):

```
package com.company;
import java.util.*;
import java.io.*;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();

        for (int i = 0; i < number; i++) {
            int numbers = input.nextInt();
            int[] arr = new int[numbers];
            for (int j = 0; j < numbers; j++) {
                arr[j] = input.nextInt();
            }
            Arrays.sort(arr);
            if (numbers % 2 == 0) {
                for (int z = 0; z < numbers / 2; z++)
{
      System.out.print(arr[(numbers / 2) + z] + " ");
System.out.print(arr[(numbers / 2) - 1 - z] + " ");
                }
                System.out.println(" ");
            }else {
                for (int z = 0; z < numbers / 2; z++)
{
                    System.out.print(arr[(numbers /
2) + z] + " ");
                    System.out.print(arr[(numbers /
2) - 1 - z] + " ");
                }
                System.out.print(arr[numbers-1]+" ");
                System.out.println(" ");
            }
        }
    }
}
```

Luke Skywalker gave Chewbacca an integer number *x*. Chewbacca isn't good at numbers but he loves inverting digits in them. Inverting digit *t* means replacing it with digit 9 - *t*.

Help Chewbacca to transform the initial number *x* to the minimum possible positive number by inverting some (possibly, zero) digits. The decimal representation of the final number shouldn't start with a zero.

Input

The first line contains a single integer *x* ($1 \le x \le 10^{18}$) — the number that Luke Skywalker gave to Chewbacca.

Output

Print the minimum possible positive number that Chewbacca can obtain after inverting some digits. The number shouldn't contain leading zeroes.

```
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {
    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        String number = input.nextLine();

        for (int i=0;i<number.length();i++) {

            int digit =
Integer.parseInt(String.valueOf(number.charAt(i)));

            if (digit>=5){
                if (digit==9 && i==0){
                    System.out.print('9');
                }else {
                    System.out.print(9-digit);
                }
            }else {
                System.out.print(digit);
            }
        }
    }
}
```

You are standing on the OX-axis at point 0 and you want to move to an integer point $x>0$. You can make several jumps. Suppose you're currently at point y (y may be negative) and jump for the k-th time. You can:

- either jump to the point $y+k$ or jump to the point $y-1$.

What is the minimum number of jumps you need to reach the point x?

Input

The first line contains a single integer t ($1 \le t \le 1000$) the number of test cases.

The first and only line of each test case contains the single integer x ($1 \le x \le 10$^6) the destination point.

Output

For each test case, print the single integer — the minimum number of jumps to reach x. It can be proved that we can reach any integer point x.

Answer(java):

```java
package com.company;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int z=0;
        int steps=0;
        for (int i=0;i<number;i++){
            int numbers= input.nextInt();

            while(steps*(steps+1)<2*numbers){
                steps++;
            }
            if (steps*(steps+1)/2==numbers+1){
                steps++;

            }
            System.out.println(steps);
            steps=0;
        }
    }
}
```

You are given a string s of even length n. String s is binary, in other words, consists only of 0's and 1's. String s has exactly $n2$ zeroes and $n2$ ones (n is even). In one operation you can reverse any substring of s. A substring of a string is a contiguous subsequence of that string. What is the minimum number of operations you need to make string s alternating? A string is alternating if $s_i \neq s_{i+1}$ for all i. There are two types of alternating strings in general: 01010101... or 10101010...

Input

The first line contains a single integer t ($1 \leq t \leq 1000$) the number of test cases.

The first line of each test case contains a single integer n ($2 \leq n \leq 10$^5; n is even) the length of string s.

The second line of each test case contains a binary string s of length n ($s_i \in$ {0, 1}). String s has exactly $n2$ zeroes and $n2$ ones.

It's guaranteed that the total sum of n over test cases doesn't exceed 10^5.

Output

For each test case, print the minimum number of operations to make s alternating.

6 1300 SCORE PROBLEMS

The Little Girl loves problems on games very much. Here's one of them.

Two players have got a string *s*, consisting of lowercase English letters. They play a game that is described by the following rules:

- The players move in turns; In one move the player can remove an arbitrary letter from string *s*.
- If the player before his turn can reorder the letters in string *s* so as to get a palindrome, this player wins. A palindrome is a string that reads the same both ways (from left to right, and vice versa). For example, string "abba" is a palindrome and string "abc" isn't.

Determine which player will win, provided that both sides play optimally well — the one who moves first or the one who moves second.

Input

The input contains a single line, containing string *s* ($1 \le |s| \le 10^3$). String *s* consists of lowercase English letters.

Output

In a single line print word "First" if the first player wins (provided that both players play optimally well). Otherwise, print word "Second". Print the words without the quotes.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        String word = input.nextLine();
        char[]words=new char[word.length()];
        int []sums=new int[word.length()+1];
        int sum=1;
        for (int i = 0; i < word.length(); i++) {
            words[i]=word.charAt(i);
        }
        Arrays.sort(words);
        for (int j=0;j<word.length()-1;j++){
            if (words[j]==words[j+1]){
                words[j]=0;
                sum=sum+1;
            }else {
                sums[j]=sum;
                sum=1;
            }
            if (j+1==word.length()-1){
                sums[word.length()]=sum;
            }
        }
        int result=0;
        for (int k=0;k< word.length()+1;k++){
            if (sums[k]%2==1){
                result =result+1;
            }
        }
        if (result%2==0 && result!=0){
            System.out.println("Second");
        }else {
            System.out.println("First");
        }
    }
}
```

Little Susie listens to fairy tales before bed every day. Today's fairy tale was about wood cutters and the little girl immediately started imagining the choppers cutting wood. She imagined the situation that is described below.

There are n trees located along the road at points with coordinates $x_1, x_2, ..., x_n$. Each tree has its height h_i. Woodcutters can cut down a tree and fell it to the left or to the right. After that it occupies one of the segments $[x_i - h_i, x_i]$ or $[x_i;x_i + h_i]$. The tree that is not cut down occupies a single point with coordinate x_i. Woodcutters can fell a tree if the segment to be occupied by the fallen tree doesn't contain any occupied point. The woodcutters want to process as many trees as possible, so Susie wonders, what is the maximum number of trees to fell.

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the number of trees.

Next n lines contain pairs of integers x_i, h_i $(1 \le x_i, h_i \le 10^9)$ — the coordinate and the height of the i-th tree.

The pairs are given in the order of ascending x_i. No two trees are located at the point with the same coordinate.

Output

Print a single number the maximum number of trees that you can cut down by the given rules.

Answer(java):

```java
package com.company;
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;

public class Main {

    public static void main(String args[]) throws
IOException {

        Scanner input = new Scanner(System.in);
        int number = input.nextInt();
        int [] x=new int[number];
        int [] h=new int[number];
        int [] dp=new int[number];
        int sum=0;


        for (int i=0;i<number;i++){
            x[i]= input.nextInt();
            h[i]= input.nextInt();

        }
        for (int j=0;j<number;j++) {

            if (j==0){
                dp[j]=x[j];
                sum++;
            }else if (x[j] - h[j] > dp[j - 1]){
                dp[j]=x[j];
                sum++;
    }else if (j<number-1 && x[j] + h[j] < x[j + 1]){
                dp[j]=x[j]+h[j];
                sum++;
            }else if (j==number-1){
                dp[j]=x[j];
                sum++;
            }else {
                dp[j]=x[j];
            }
        }
        System.out.println(sum);
    }
}
```

ABOUT THE AUTHOR

He is a computer science student at the University of Debrecen in Hungary. Most of his projects are in Java, but some others are in python and C. His favorite subject is Machine Learning, and he have some publications and projects in this field. He also have some other experience in other subjects such as android studio and HTML. He doesn't want to say that he is one of the genius people in the world, but he think he is the most hard-working one.