FAU-Programmierwettbewerb 2003

28. Juni 2003

Problemübersicht:

Viel Spaß beim Programmieren!

Problem A
The ? 1 ? 2 ? ... ? n = k Problem

Author: Alex Gevak

Given the following formula, one can set operators '+' or '-' instead of each '?', in order to obtain a given k: *? 1 ? 2 ? ... ? n = k*.

For example, to obtain $k = 12$, the expression to be used will be:

$$-1 + 2 + 3 + 4 + 5 + 6 - 7 = 12,$$

with $n = 7$.

Input

In the first line of the input file, there is an integer number c, indicating the number of test cases that will follow. In each of the the following c lines, there will be an integer number $k : 0 \leq |k| \leq 1000000000$.

Output

For each test case, your program has to print a single line containing the minimal possible integer n such that the above equation can be satisfied. Please note that the left-hand side of the equation *must not* be empty, i.e. $n \geq 1$.

Sample Input

```
2
12
-3646397
```

Sample Output

```
7
2701
```

Problem B
A Ferrari (but a fast one)

Author: Stefan Büttcher

A man named "Shoe" has recently shot down the `faui01`. Now somebody tells him that he has got two possibilities:

- going to the CIP admins and confessing, or
- buying a Ferrari, but a fast one.

Since Shoe has been a coward since birth, he decides to buy a Ferrari and run away. Unfortunately for him, the car he buys from the dealer (the deal has to be made very fast) should be delivered to the US market, so the fuel consumption display is in miles/gallon only, which Shoe does not understand. For some reason we don't know it is very important for Shoe to know what his automobile's fuel consumption is, and he has to know it in litres/kilometer. In addition, he has to know how many kilometers he can still go until refueling the next time.

In order to make his escape successful, it is your task to translate the display messages of the Ferrari's board computer for him.

For all your calculations, assume that 1 mile is 1609 meters, while 1 gallon is 3.8 litres. Assume that the Ferrari has 25 gallons of gasoline at the beginning of *every* test case.

Input

Input consists of a number of lines (test cases) each of which contains two floating-point numbers x and y, where x is the distance (in miles) driven so far and y is the fuel consumption (in miles/gallon). Input is terminated by an empty line or EOF.

Output

For each test case, your program must print a line containing the fuel consumption (as litres per 100 kilometers), rounded to the nearest integer and the distance left to go until the car is out of gas, rounded in the same way.
Please note that in the Sample Output is does not say "231/100km" but "23l/100km" (with a small "L", as we are talking about litres per kilometer).

Sample Input

```
231.1 10.3
175 7
```

Sample Output

```
Fuel consumption: 23l/100km. Next stop: 42km.
Fuel consumption: 34l/100km. Next stop: 0km.
```

Problem C
Bi-Primes

Author: Stefan Büttcher

Every positive integer which is only divisible by 1 and itself is called a prime. Accordingly, a bi-prime is a number k that is prime and whose inverse is also prime. By inverse we mean the integer number that results when the order of k's digits is reversed.

$$isBiPrime(n) \Leftrightarrow isPrime(n) \wedge isPrime(inv(n)).$$

For example, $inv(107) = 701$. Since both 107 and 701 are primes, 107 is a bi-prime. For integers $k < 10$, we obviously have $k = inv(k)$, so all primes smaller than 10 are automatically bi-primes. When inverting a number, leading zeros must be ignored: $inv(10200) = 201$, but $inv(201) = 102$. Hence, we cannot be sure that $inv(inv(k)) = k$.

Input

Input consists of a number of lines, each of which contains a positive integer $n < 10^9$. Input is terminated by an empty line or EOF.

Output

For each input line, print one output line which contains n if n is a bi-prime or the next bigger bi-prime number if n is no bi-prime.

Sample Input

```
1
2
15
48
130
7312876
20802187
```

Sample Output

```
2
2
17
71
131
7312909
30000037
```

Problem D
Complete Surveillance

Author: Dominik Scheder

Shak Shi Rakh, president of a small totalitarian dictatorship, has a great plan: he wants to put his citizens under complete surveillance. To achieve his goal, he is going to found a huge intelligence service. He wants that every citizen that is not a member of the service itself must have at least one friend working in the service, so that this one can observe him (of course he won't know that his friend is observing him). Also, no two members of the service should be friends, since this would imply the danger of a conspiracy.

Thus, it is not clear at all if Shak Shi Rakh can achieve his ambitious goal. Since you are his top computer scientist, your task is, given a list of the citizens indicating which are friends, to determine if Shak Shi Rakh can build a service as described above. If not, he will be forced to have some citizens executed, but this won't be part of your task.

You should be careful in designing your algorithm since the number of citizens in the country can be as large as 100. You can assume that friendship is a symmetric relation, i.e. a is friends with b if and only if b is friends with a.

Input

The first line of the input contains a single integer c, the number of test cases following. The first line of each test case contains two integers v, e, the number of citizens and friendship relations in the country. The next e lines each consist of two integers i, j: $0 \leq i, j < v$, indicating that i and j are friends.

Output

For each input set, simply output "Yes." if Shak Shi Rakh can put his citizens under the perfect surveillance described above, or "No." if he cannot achieve this without having some citizens executed.

Sample Input

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

Sample Output

```
Yes.
```

Problem E
Cool People are always late

Author: Stefan Büttcher

As you might know, the cool people are never on time when they join a party. Instead, they prefer to be among the last that arrive at the party. That makes them important.

Today, you want to be important. Unfortunately, you are quite bored just sitting at home and doing nothing because your computer is in repair. So, you eventually decide to leave your apartment. The only problem is that you would definitely not like to be the first to arrive at the place, since that would show everybody that you have got no friends and are a really miserable being. On the other hand, you do not want to be too late, either, because you are not cool enough yet.

Your task is, given a map from your city, to find the second-shortest route from your home to the party. This is because the shortest route would let you arrive too early, while taking any route longer than the second-shortest is still too cool for you. You will, of course, not have any cycles in your route, because that would be really uncool (a cycle in this sense is any route on which the same place is visited more than once).

It is guaranteed that there is exactly one route of shortest length.

Input

The first line of the input file contains a single integer n, the number of test cases that will follow. The next line is blank.

For each test case, there is one line with two integers v and e, the number of places (vertices) on the map and the number of streets (edges) connecting them, respectively. The next e lines contain the description of a street, consisting of three integers a, b and len, where $0 \leq a, b < v$. Thus, the line "3 8 10" would mean that there is a street of length 10 connecting the places 3 and 8. Each test case is finished by a line with two integers h and p, the place of your home and the one of the party, followed by a blank line.

There are no one-way streets, and for every pair of places (a, b) there is at most one street connecting them. No street has a negative length, and no street is longer than 1000. The number of places will not exceed 100, the number of streets will not exceed $1,000$.

Output

For every test case, print a line with a description of the second-shortest path from your home to the party. If there is no second-shortest route, print "I have to be uncool". If there is more than just one second-shortest route, print "Multiple possible paths of length k", where k is the length of the second-shortest paths. The exact format can be seen in the examples below.

Sample Input

```
3

4 4
0 1 10
0 3 5
1 2 10
3 1 8
0 2

4 3
0 1 10
1 2 10
2 3 10
0 3
```

```
5 6
0 1 8
0 3 5
1 2 7
1 4 5
3 1 5
4 2 4
0 2
```

Sample Output

```
Path of length 23 found: 0->3->1->2
I have to be uncool
Multiple possible paths of length 17
```

Problem F
Geometry sucks

Author: Stefan Büttcher

Everybody knows that most geometrical problems, posed in programming contests, suck. However, this is a really easy problem. Given an arbitrary polygon and a point, you have to determine if the point lies within the polygon or not (in our terminology, the border of a polygon belongs to its inside, which means that e.g. all vertices of a polygon lie within it).

Input

Input consists of a number of test cases, each of which consists of two lines. The first line contains the description of the polygon:

$$n\ x_1\ y_1\ x_2\ y_2\ \ldots\ x_n\ y_n.$$

n is the number of points that make up the polygon, and all the (x_j, y_j) pairs are the descriptions of the n points. The polygon's points are given in their "natural" order. The second line of every test case contains the description of the point to be examined: $x_p\ y_p$.

Output

For each test case, print YES if the point lies within the polygon and NO otherwise.

Sample Input

```
4 0 0 1 0 1 1 0 1
0.5 0.5
3 1 2 5 2 3 4
0 0
10 2 1 4 2 5 1 6 4 4 5 3 4 2 6 1 4 2.7 3 1 2
2.0 4.0
10 2 1 4 2 5 1 6 4 4 5 3 4 2 6 1 4 2.7 3 1 2
2.0 3.0
```

Sample Output

```
YES
NO
YES
NO
```

Problem G
Three Mathematicians

Author: Stefan Büttcher

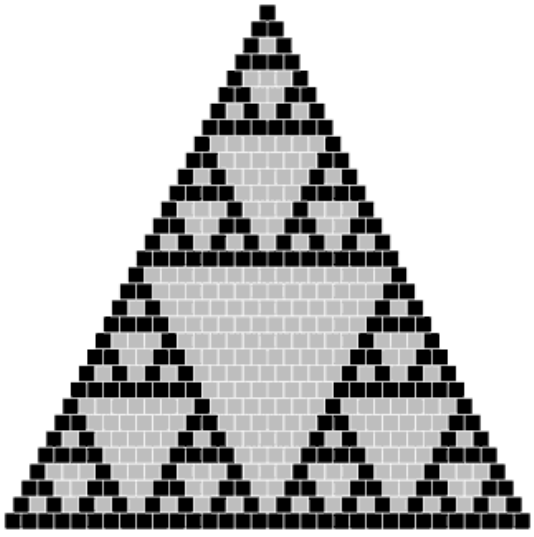
Everybody knows the Pascal Triangle from school. It is named after the French mathematician Blaise Pascal (1623 - 1662). Back in school, when nobody used to be familiar with factorials 'n' stuff, the triangle helped us calculating binomial coefficients. For those who don't remember, here comes the upper part of the triangle:

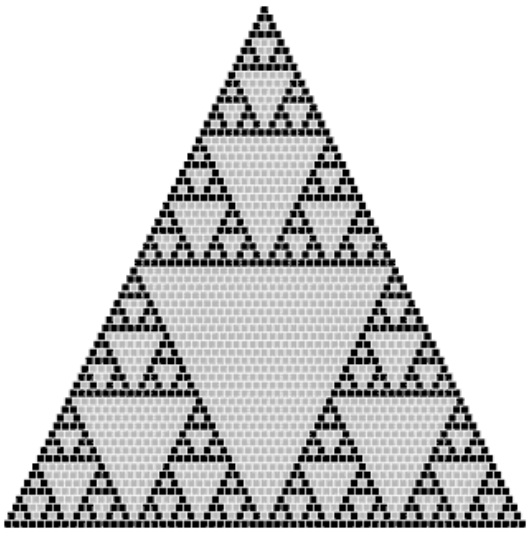
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

Apart from its mathematical importance, the Pascal Triangle is also a part of modern art. You can create a wonderful picture by painting all numbers occurring inside the triangle either red or blue, depending on if they are odd or even. Thus, the upper part of the triangle would look like this:

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

The Polish mathematician Waclaw Sierpinski (1882 - 1969) studied the properties of this coloured triangle (this is why it also sometimes called Sierpinski Triangle) and found out that it is fractal, i.e. there are parts of the triangle that are very similar to the entire triangle. You can see this when looking at the following two pictures showing the Sierpinski Triangle for $n = 32$ and $n = 64$.





Many people are really ecstatic about the beauty of the triangle. One of them is the famous mathematician Hans Grabmüller. Since his retirement, he plans to paint a very large version of the Sierpinski Triangle and put it into his living room. Unfortunately, Grabmüller has no idea how many of the numbers inside the triangle are odd and how many are even. So, he does not know how much ink he has to buy. You will assist him with calculating the amount of ink needed to paint the triangle.

Input and Output

Input consists of n lines, each of which contains a number $j < 4000$. For every line of input, print one line of output containing o and e, the number of odd and even elements in the j'th line of the triangle.

Sample Input

```
0
1
2
3
4
5
6
7
```

Sample Output

```
1 0
2 0
2 1
4 0
2 3
4 2
4 3
8 0
```

Problem H
Washing the Dishes

Author: Dominik Scheder

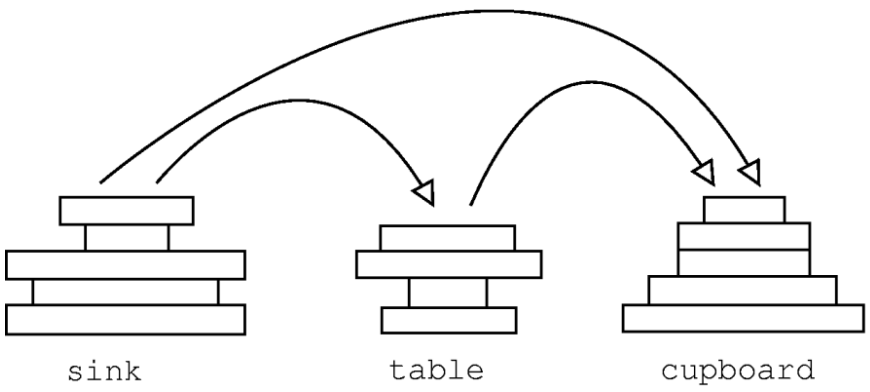
The little Hans has just moved from Nürnberg to a *WG* (flat share) in Erlangen, where he studies Computer Science. Moving out of your parents house brings a lot of fun, but, as life goes, also a lot of new work.

Today, the Hans is washing the dishes. Unfortunately, this has not been done for 3 months, and there are only three places left in the kitchen for the dishes:

- They can be in the sink, where they all actually are at the beginning,
- they can be at the one spot on the table not covered by pizza, empty beer bottles or other relicts of the fun the Hans and his friends had the last days,
- they can be in the cupboard, where they eventually *should* be.

But, look, the sink, the table and the cupboard are all so crowded that no two dishes can be put side by side on any of these three places: the Hans has to stack them and can only move the topmost dish of a stack onto the top of another stack! Moreover, the dishes have different sizes, and the Hans wants dishes in the cupboard to be ordered by their size.
If you find the Hans' task has some similarity with the *Towers of Hanoi*, you are not at all wrong! But there are two little differences between the towers and the Hans' task:

- the dishes in the sink and on the table can be in any order, i.e. there may be bigger dishes on top of smaller ones. Only in the cupboard should they be ordered.
- unlike in *Towers of Hanoi*, the Hans can only move dishes *forward*, i.e. from the sink to the table, from the table into the cupboard or directly from the sink into the cupboard.



So, it is not obvious at all if the Hans can move all dishes into the cupboard. Your job is to write a program, that, given the sizes of the dishes originally in the sink, determines if the Hans can fulfill his task or better should go back to his computer instead of wasting his time doing an impossible job.

Input

The first line of the input contains a single integer c, the number of test cases following. Each test case consists of a single line of the form $n\ a_1\ a_2\ \ldots\ a_n$, where n is the number of dishes in the sink, and a_i is the size of the ith dish, numbered from the bottom of the sink to the top.

Output

For each test case, if the Hans can fulfill his task, print "Go on, Hans!", else print "Go back to your Computer, Hans!"

Sample Input

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

Sample Output

```
Go on, Hans!
Go back to your computer, Hans!
```