INTERNATIONAL TOURNAMENT IN INFORMATICS
Shumen, 24 November 2012, Senior Group

Task A1. TELEPORT

The software company, where the young programmer Stancho was hired, is famous with its distributed software for management of small planetary systems, the number of planets of which is a power of two. Recently, the company issued the current version of the software and the young professionals of the company have to visit the clients and to install the updates. Stancho also has to visit the planets of one system-customer. Traveling from one planet to another planet of the system is not easy. Imagine that the planets are ordered in line by their distances to the star of the system and labeled with the numbers 0, 1, …, $2^n–1$ following this order (astronomically this is rare, but not impossible). Stancho has $2^n–1$ teleports, labeled with 1, 2,…,$2^n–1$. The teleport labeled with t could transmit only once one human from a planet labeled with m to the planets with label $m+t$ or label $m–t$, if such a planet exists (i.e. if $0 \le m–t$ and $m+t \le 2^n–1$). Using a space-stop Stancho could go to the planet labeled with k. Write a program **`teleport`**, which for given n and k has to find the best way Stancho can use the teleports in order to visit as many as possible planets of the system.

Input

On the single line of the standard input the positive integers, n and k are given, $1 \le n \le 20$, $0 \le k \le 2^n–1$.

Output

On the first line of the standard output the program has to print the maximal number p of planets that Stancho could visit – excluding the planet k. On the next row, the program has to print a sequence of the numbers of p teleports that Stancho has to use, in the order of using. The numbers of teleports that transmit him from a planet with larger number to a planet with smaller number has to be multiplied by –1. If there are several different ways to do the teleporting the program has to print any one of them.

Constraints

$1 \le n \le 20$, $0 \le k \le 2^n–1$.

Example

Input	**Possible output**

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

Task A2. THE WIDEST BOATS

In Waterland, there are *N* lakes (numbered from 1 to *n*) and *M* channels between them. The width *Wi,j* (in meters) of each channel *(i,j)* is known. Navigation in the channels can be performed in both directions.

A transportation company has prepared a list of *K* pairs of lakes, among which will be held regular services with boats of people and goods. There exists a path connecting any two lakes in a pair of the list, but not always this path is a direct one, i.e. this path may pass thru other lakes.

Write program **boats** that calculates the maximum width of boats, which can pass between the pairs of lakes in the list (a boat can move from one lake to another, if its width is less than or equal to the width of the channel, connecting the lakes).

Input

On the first line of the standard input are given three integers *N, M* and *K.*

On each one of the next *m* lines are given three integers *i*, *j* and *w*, showing that there is a channel of width *w* between lakes *i* and *j* ($1 \leq i, j \leq N$).

There follow *K* rows, each containing two lakes' numbers *i* and *j*, between which a boat of the company will travel.

Output

The program should write *K* rows on the standard output, each containing one integer, equal to the maximum width of the boat that can travel between the corresponding two lakes.

Constraints

$N \leq 1000$, $M \leq 100000$, $K \leq 10000$, $Wi,j \leq 200$

Example

Input

```
6 9 4
1 2 2
1 4 3
1 6 1
2 3 3
2 5 2
3 4 4
3 6 2
4 5 5
5 6 4
2 6
3 5
1 2
4 6
```

Output

```
3
4
3
4
```

Problem A3. LINES

You are given *N* rays (half-lines) with initial points on the Oy axis. None of the rays is parallel to the Oy axis. The rays are given in the form $y = Ai*x + Bi$. Each ray is only defined for positive x.

Write a program `lines` that answers *Q* queries of the type: What is the maximal x-coordinate of an intersection point of the line $y = Cj*x + Dj$ with one of the given *N* rays.

Input

On the first line of the standard input is given a single positive integer *N*, the number of the rays.

On each of the next *N* lines are given the two integers: *Ai* and *Bi*, the coefficients of the equation of each ray.

On the next line a single positive integer *Q* is given, the number of queries.

On each of the following *Q* lines are given two integers *E* and *F*. If the previous line had crossed one of the rays, or if this is the first line, these are the coefficients of the equation of the current line ($Cj = E$, $Dj = F$). Otherwise, the coefficients of the equation of the line are $Cj = E$ ^ $(2^{29}-1)$ and $Dj = F$ ^ $(2^{29}-1)$, where ^ is the XOR operation.

Output

For each question output a single real number x with at least 6 digits after the decimal point, the maximal x-coordinate of a point in which the current line crosses one of the given rays. If the line doesn't cross any ray, output "No cross".

Example

Input

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

Output

```
0.75000000
No cross
1.00000000
```

Constraints

All the numbers in the input are integers.

$-1000000000 < Ai,Bi,Ci,Di < 1000000000$

All numbers *Ai* are different.

Each *Cj* is different from any of the numbers *Ai*.

Each *Dj* is different from any of the numbers *Bi*.

Subtask 1 (15 points)

$0 < N, Q < 2001$

Subtask 2 (20 points)

$0 < N, Q < 30001$; $Cj = 0$ for each *j*.

Subtask 3 (25 points)

$0 < N$, $Q < 30001$; Each line crosses at least one of the rays; At least one *Cj* is not 0.

Subtask 4 (30 points)

$0 < N$, $Q < 30001$

Subtask 5 (10 points)

$0 < N$, $Q < 50001$

Grading

Points for each subtask will be received only if your program solves correctly **all** test cases given for that particular subtask.