Problem A

Crashing Robots

In a modernized warehouse, robots are used to fetch the goods. Careful planning is needed to ensure that the robots reach their destinations without crashing into each other. Of course, all warehouses are rectangular, and all robots occupy a circular floor space with a diameter of 1 meter. Assume there are N robots, numbered from 1 through N. You will get to know the position and orientation of each robot, and all the instructions, which are carefully (and mindlessly) followed by the robots. Instructions are processed in the order they come. No two robots move simultaneously; a robot always completes its move before the next one starts moving.

A robot crashes with a wall if it attempts to move outside the area of the warehouse, and two robots crash with each other if they ever try to occupy the same spot.

Input specifications

The first line of input is K, the number of test cases. Each test case starts with one line consisting of two integers, $1 \leq A, B \leq 100$, giving the size of the warehouse in meters. A is the length in the EW-direction, and B in the NS-direction.

The second line contains two integers, $1 \leq N, M \leq 100$, denoting the numbers of robots and instructions respectively.

Then follow N lines with two integers, $1 \leq X_i \leq A, 1 \leq Y_i \leq B$ and one letter (`N`, `S`, `E` or `W`), giving the starting position and direction of each robot, in order from 1 through N. No two robots start at the same position.

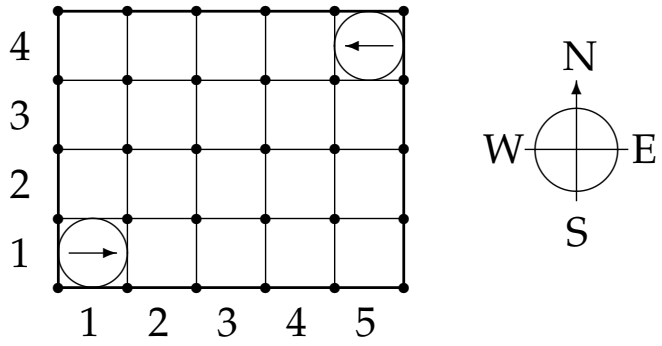


Figure 1: The starting positions of the robots in the sample warehouse

Finally there are M lines, giving the instructions in sequential order. An instruction has the following format:

`<robot #> <action> <repeat>`

Where `<action>` is one of

- `L`: turn left 90 degrees,
- `R`: turn right 90 degrees, or
- `F`: move forward one meter,

and $1 \leq$ `<repeat>` ≤ 100 is the number of times the robot should perform this single move.

Output specifications

Output one line for each test case:

- `Robot i crashes into the wall`, if robot i crashes into a wall. (A robot crashes into a wall if $X_i = 0$, $X_i = A + 1$, $Y_i = 0$ or $Y_i = B + 1$.)
- `Robot i crashes into robot j`, if robots i and j crash, and i is the moving robot.
- `OK`, if no crashing occurs.

Only the first crash is to be reported.

Sample input

```
4
5 4
2 2
1 1 E
5 4 W
1 F 7
2 F 7
5 4
2 4
1 1 E
5 4 W
1 F 3
2 F 1
1 L 1
1 F 3
5 4
```

```
2 2
1 1 E
5 4 W
1 L 96
1 F 2
5 4
2 3
1 1 E
5 4 W
1 F 4
1 L 1
1 F 20
```

Output for sample input

```
Robot 1 crashes into the wall
Robot 1 crashes into robot 2
OK
Robot 1 crashes into robot 2
```

Problem B

Funny Games

Nils and Mikael are intergalaxial fighters. Now they are competing for the planet Tellus. The size of this small and unimportant planet is $1 < X < 10000$ gobs. The problem is that their pockets only have room for one gob, so they have to reduce the size of the planet. They have available $1 \leq K \leq 6$ FACTOR-weapons characterized by numbers $F_1, F_2, \ldots, F_k$, all less than 0.9. As is commonly known, a FACTOR-weapon will blow off part of the planet, thus reducing the planet to a fraction of its size, given by the characteristic. Thus, with e.g. $F_1 = 0.5$ an application of the first weapon will half the size of the planet. The fighter who reduces the size to less than, or equal to, 1 gob can take the planet home with him. They take turns attacking the planet with any weapon. If Nils starts, who will win the planet? Assume that both Nils and Mikael are omniscient and always make a winning move if there is one.

Technical note: To ease the problem of rounding errors, there will be no edge cases where an infinitesimal perturbation of the input values would cause a different answer.

Input specifications

The first line of input is $N \leq 100$, the number of test cases. Each of the next N lines consists of X, K and then the K numbers $F_1, F_2, \ldots, F_k$, having no more than 6 decimals

Output specifications

For each test case, produce one line of output with the name of the winner (either Nils or Mikael).

Sample input

```
4
6 2 0.25 0.5
10 2 0.25 0.5
29.29 4 0.3 0.7 0.43 0.54
29.30 4 0.3 0.7 0.43 0.54
```

Output for sample input

```
Mikael
Nils
Nils
Mikael
```

Problem C

Nullary Computer

Decimal	Nullary
0	
1	0
2	00
3	000
4	0000
5	00000
...	...

Brian Huck has invented a new power-saving computer. With the current CMOS-based processors, a certain amount of power is lost each time a bit is changed from 0 to 1 or back. To avoid this problem, Brian's new Nullary Core stores only zeros. All numbers are stored in nullary form, as shown on the right.

His initial 64-nit model has 26 registers, each of which may store up to 64 nits, and any attempt to store more than 64 nits will result in a run time error. There is also a flag register, which contains either a zero, or nothing. The instruction set is as follows:

Table 1: NC Instruction Set

Instruction	Explanation and how to simulate in C
`A`	Add a zero to the value in register A (similarly for all uppercase letters). `a++;`
`a`	First, empty the flag register. Then, if possible, remove a zero from register A, and place it in the flag register (similarly for all lowercase letters) `flag = 0; if(a>0) { flag=1; a--; }`
`(`	If the flag register is empty, jump past the matching `)`. Otherwise, empty the flag register. `while(flag) { flag=0;  ....`
`)`	Jump to the matching `(`. `... }`

Apart from instructions, no other characters than whitespace are allowed in a nullary program.

Sample programs

Brian has provided some programs to illustrate the elegance and simplicity of his computer.

Your task will be to write a sorting program for Brian's Nullary Core-based Prototype Computer. The NCPC has limited memory, so your

Table 2: Sample NC programs

`b(b)a(Ba)`	Move register A to register B (by first emptying register B, then repeatedly pulling a single zero from register A and placing it into B).
`XXXa(GIa)i(g(FY g)y(Gy)f(Zb(z)z (i(YBi)y(Iy))f) Zb(zb)z(xz)i)x`	Set the flag register if the number of zeros in register A is prime.

program must be no longer than 5432 instructions. Also, the running time of your program must be no more than $5 \cdot 10^6$ steps for any possible input, where a step is considered to be the execution of one instruction.

Important note: You must submit the nullary source code of this program, and not some Java, C or C++ source code.

Input specifications

The numbers to be sorted will be given in the first 24 registers `A`-`X`; the remaining two registers (`Y` and `Z`) will be empty.

Output specifications

The sorted numbers should be in registers `A` through `X`, in increasing order. Register Y and Z should be empty.

Sample input

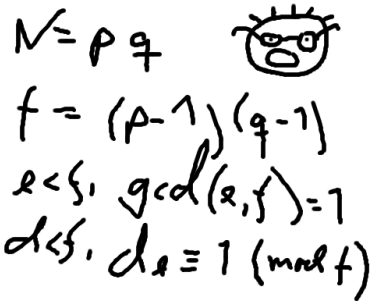
A 0	J 000	S
B 000000000	K	T
C 000000	L	U
D 0000	M	V
E 00000000	N	W
F 0000000	O	X 0
G 0000	P	Y
H 000000	Q	Z
I 000000000	R	

Output for sample input

A	J	S 000000
B	K	T 000000
C	L	U 0000000
D	M	V 00000000
E	N 0	W 000000000
F	O 0	X 000000000
G	P 000	Y
H	Q 0000	Z
I	R 0000	

Problem D

The Embarrassed Cryptographer



The young and very promising cryptographer Odd Even has implemented the security module of a large system with thousands of users, which is now in use in his company. The cryptographic keys are created from the product of two primes, and are believed to be secure because there is no known method for factoring such a product effectively.

What Odd Even did not think of, was that *both* factors in a key should be large, not just their product. It is now possible that some of the users of the system have weak keys. In a desperate attempt not to be fired, Odd Even secretly goes through all the users keys, to check if they are strong enough. He uses his very poweful Atari, and is especially careful when checking his boss' key.

Input specifications

The input consists of no more than 20 test cases. Each test case is a line with the integers $4 \leq K \leq 10^{100}$ and $2 \leq L \leq 10^6$. K is the key itself, a product of two primes. L is the wanted minimum size of the factors in the key. The input set is terminated by a case where $K = 0$ and $L = 0$.

Output specifications

For each number K, if one of its factors are strictly less than the required L, your program should output "BAD p", where p is the smallest factor in K. Otherwise, it should output "GOOD". Cases should be separated by a line-break.

Sample input

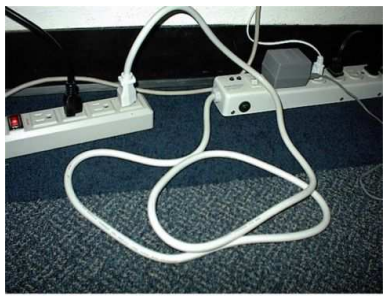
```
143 10
143 20
667 20
667 30
2573 30
2573 40
0 0
```

Output for sample input

```
GOOD
BAD 11
GOOD
BAD 23
GOOD
BAD 31
```

Problem E

Electrical Outlets



Roy has just moved into a new apartment. Well, actually the apartment itself is not very new, even dating back to the days before people had electricity in their houses. Because of this, Roy's apartment has only one single wall outlet, so Roy can only power one of his electrical appliances at a time.

Roy likes to watch TV as he works on his computer, and to listen to his HiFi system (on high volume) while he vacuums, so using just the single outlet is not an option. Actually, he wants to have all his appliances connected to a powered outlet, all the time. The answer, of course, is power strips, and Roy has some old ones that he used in his old apartment. However, that apartment had many more wall outlets, so he is not sure whether his power strips will provide him with enough outlets now.

Your task is to help Roy compute how many appliances he can provide with electricity, given a set of power strips. Note that without any power strips, Roy can power one single appliance through the wall outlet. Also, remember that a power strip has to be powered itself to be of any use.

Input specifications

Input vill start with a single integer $1 \leq N \leq 20$, indicating the number of test cases to follow. Then follow N lines, each describing a test case. Each test case starts with an integer $1 \leq K \leq 10$, indicating the number of power strips in the test case. Then follow, on the same line, K integers separated by single spaces, $O_1\ O_2 \ldots O_K$, where $2 \leq O_i \leq 10$, indicating the number of outlets in each power strip.

Output specifications

Output one line per test case, with the maximum number of appliances that can be powered.

Sample input

```
3
3 2 3 4
```

```
10 4 4 4 4 4 4 4 4 4 4
4 10 10 10 10
```

Output for sample input

```
7
31
37
```

Problem F

Worst Weather Ever

"Man, this year has the worst weather ever!", David said as he sat crouched in the small cave where we had sought shelter from yet another sudden rainstorm.

"Nuh-uh!", Diana immediately replied in her traditional know-it-all manner.

"Is too!", David countered cunningly.

Terrific. Not only were we stuck in this cave, now we would have to listen to those two nagging for at least an hour. It was time to cut this discussion short.

"Big nuh-uh. In fact, 93 years ago it had already rained five times as much by this time of year."

"Duh", David capitulated, "so it's the worst weather in 93 years then."

"Nuh-uh, this is actually the worst weather in 23 years.", Diana again broke in.

"Yeah, well, whatever", David sighed, "Who cares anyway?".

Well, dear contestants, you care, don't you?

The Problem

Your task is to, given information about the amount of rain during different years in the history of the universe, and a series of statements in the form "Year X had the most rain since year Y", determine whether these are true, might be true, or are false. We say that such a statement is true if:

- The amount of rain during these two years and all years between them is known.
- It rained at most as much during year X as it did during year Y.
- For every year Z satisfying $Y < Z < X$, the amount of rain during year Z was less than the amount of rain during year X.

We say that such a statement might be true if there is an assignment of amounts of rain to years for which there is no information, such that the statement becomes true. We say that the statement is false otherwise.

Input specifications

The input will consist of several test cases, each consisting of two parts.

The first part begins with an integer $1 \leq n \leq 50000$, indicating the number of different years for which there is information. Next follow n lines. The ith of these contains two integers $-10^9 \leq y_i \leq 10^9$ and $1 \leq r_i \leq 10^9$ indicating that there was r_i millilitres of rain during year y_i (note that the amount of rain during a year can be any nonnegative integer, the limitation on r_i is just a limitation on the input). You may assume that $y_i < y_{i+1}$ for $1 \leq i < n$.

The second part of a test case starts with an integer $1 \leq m \leq 10000$, indicating the number of queries to process. The following m lines each contain two integers $-10^9 \leq Y < X \leq 10^9$ indicating two years.

There is a blank line between test cases. The input is terminated by a case where $n = 0$ and $m = 0$. This case should not be processed.

Technical note: Due to the size of the input, the use of cin/cout in C++ might be too slow in this problem. Use scanf/printf instead. In Java, make sure that both input and output is buffered.

Output specifications

There should be m lines of output for each test case, corresponding to the m queries. Queries should be answered with "`true`" if the statement is true, "`maybe`" if the statement might be true, and "`false`" if the statement is false.

Separate the output of two different test cases by a blank line.

Sample input

```
4
2002 4920
2003 5901
2004 2832
2005 3890
2
2002 2005
2003 2005

3
1985 5782
1995 3048
2005 4890
2
1985 2005
2005 2015
```

```
0
0
```

Output for sample input

```
false
true

maybe
maybe
```

Problem G

Kingdom

King Kong is the feared but fair ruler of Transylvania. The kingdom consists of two cities and $N < 150$ towns, with nonintersecting roads between some of them. The roads are bidirectional, and it takes the same amount of time to travel them in both directions. Kong has $G < 353535$ soldiers.

Due to increased smuggling of goat cheese between the two cities, Kong has to place his soldiers on some of the roads in such a way that it is impossible to go from one city to the other without passing a soldier. The soldiers must not be placed inside a town, but may be placed on a road, as close as Kong wishes, to any town. Any number of soldiers may be placed on the same road. However, should any of the two cities be attacked by a foreign army, the king must be able to move all his soldiers fast to the attacked city. Help him place the soldiers in such a way that this mobilizing time is minimized.

Note that the soldiers cannot be placed in any of the cities or towns. The cities have ZIP-codes 95050 and 104729, whereas the towns have ZIP-codes from 0 to $N - 1$. There will be at most one road between any given pair of towns or cities.

Input specifications

The input contains several test cases. The first line of each test case is N, G and E, where N and G are as defined above and $E < 5000$ is the number of roads. Then follow E lines, each of which contains three integers: A and B, the ZIP codes of the endpoints, and ϕ, the time required to travel the road, $\phi < 1000$. The last line of the input is a line containing a single 0.

Output specifications

For each test case in the input, print the best mobilizing time possible, with one decimal. If the given number of soldiers is not enough to stop the goat cheese, print "`Impossible`" instead.

Sample input

```
4 2 6
95050 0 1
0 1 2
1 104729 1
95050 2 1
2 3 3
3 104729 1
4 1 6
95050 0 1
0 1 2
1 104729 1
95050 2 1
2 3 3
3 104729 1
4 2 7
95050 0 1
0 1 2
1 104729 1
95050 2 1
2 3 3
3 104729 1
2 1 5
0
```

Output for sample input

```
2.5
Impossible
3.0
```

Problem H

Necklace Decomposition

The set of cyclic rotations of a string are the strings obtained by embedding the string clockwise on a ring, with the first character following on the last, starting at any character position and moving clockwise on the ring until the character preceeding the starting character is reached. A string is a necklace if it is the lexicographically smallest among all its cyclic rotations. For instance, for the string 01011 the cyclic rotations are (10110,01101,11010,10101,01011), and furthermore 01011 is the smallest string and hence, a necklace.

Any string S can be written in a unique way as a concatenation $S = T_1 T_2 \dots T_k$ of necklaces T_i such that $T_{i+1} < T_i$ for all $i = 1, \dots, k-1$, and $T_i T_{i+1}$ is not a necklace for any $i = 1, \dots, k-1$. This representation is called the necklace decomposition of the string S, and your task is to find it.

The relation $<$ on two strings is the lexicographical order and has the usual interpretation: $A < B$ if A is a proper prefix of B or if A is equal to B in the first $j-1$ positions but smaller in the jth position for some j. For instance, $001 < 0010$ and $1101011 < 1101100$.

Input specifications

On the first line of the input is a single positive integer n, telling the number of test scenarios to follow. Each scenario consists of one line containing a non-empty string of zeros and ones of length at most 100.

Output specifications

For each scenario, output one line containing the necklace decomposition of the string. The necklaces should be written as '(' necklace ')'.

Sample input

```
5
0
0101
0001
```

```
0010
11101111011
```

Output for sample input

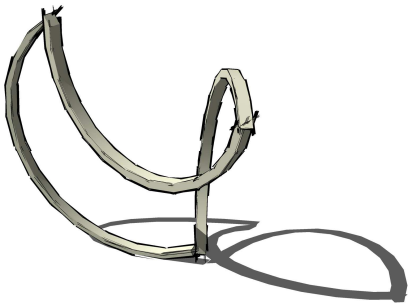
```
(0)
(0101)
(0001)
(001)(0)
(111)(01111)(011)
```

Problem I

Playground

George has $K \leq 20$ steel wires shaped in the form of half-circles, with radii $a_1, a_2, \ldots, a_K$. They can be soldered (connected) at the ends, in any angle. Is it possible for George to make a closed shape out of these wires? He does not have to use all the wires.

The wires can be combined at any angle, but may not intersect. Beware of floating point errors.



Input specifications

Each data set consists of a number $0 < K \leq 20$ on a line by itself, followed by a line of K space-separated numbers a_i. Each number is in the range $0 < a_i < 10^7$, and has at most 3 digits after the decimal point.

The input will be terminated by a zero on a line by itself.

Output specifications

For each test case, there should be one word on a line by itself; "YES" if it is possible to make a simple connected figure out of the given arcs, and "NO" if it isn't.

Sample input

```
1
4.000
2
1.000 1.000
3
1.455 2.958 4.424
7
1.230 2.577 3.411 2.968 5.301 4.398 6.777
0
```

Output for sample input

```
NO
YES
NO
YES
```