Baking Rivalry

photo from flickr

You are handling the register at the school bake sale. Cathy and Sally are two moms selling their baked goods today. At the end of the bake sale, they approach you and ask you to tell them whose cookies were better. To be unbiased, you decide rather than taste the cookies, you will look at the sequence of sales. A sale by Cathy will appear as a C and a sale by Sally will appear as an S. Whoever has more sales is the better baker!

Input Format

The input is a single line containing a string of 'S's and 'C's.

Constraints

The length of the sequence is at most 100,000.

Output Format

If Cathy won, print "Cathy won!". If Sally won, print "Sally won!" If there is a tie, print "Tie!"

Sample Input 0

```
SSC
```

Sample Output 0

```
Sally won!
```

Explanation 0

Sally had 2 sales and Cathy had 1.

Sample Input 1

```
CCSSC
```

Sample Output 1

```
Cathy won!
```

Explanation 1

Cathy had 3 sales and Sally had 2.

Sample Input 2

```
SSCC
```

Sample Output 2

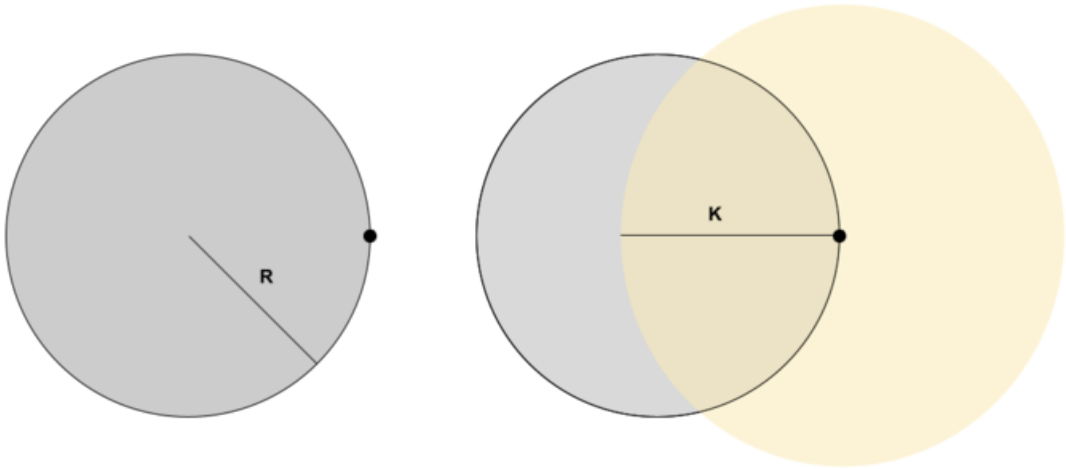
```
Tie!
```

Explanation 2

Both Cathy and Sally had 2 sales, so it is a tie.

Lighting the Way

You have been sold a powerful lamp that is said to illuminate half of any circular foggy area from the edge. However, you find this rather strange… No one measures the strength of a lamp by area relative to a fog. If this claim is true, you want to know just how far your light can travel radially. Mathematically, given a circle of radius R (the fog), you want to find the value of K, the radial distance of the light such that the overlap between the fog and your light is equal to the area of fog that is not covered by your light.



Input Format

A single line containing a float R, representing the radius of the fog

Constraints

$0 < R \leq 100$

Output Format

A single line containing a float K, representing the radial distance of the light. Precision for correctness is checked to be within 1e-3.

Sample Input 0

```
1.0
```

Sample Output 0

```
1.158728
```

Weird Programming Contest

Photo from ICPC World Finals 2014, taken from Petr's Blog

You are participating in a different kind of programming contest. There are N problems in total but you are assigned problem S only. You need to solve only the problem assigned to you to win this contest. But you find out that the only problem you can solve is problem D. Fear not, there are also M swap options. A swap option looks like the following:

$u\ v\ w$

This means that if you are currently assigned problem u, you can swap it with problem v. But at the same time, w time penalty will be added to your total time penalty. It is possible that the value of w is negative, in which case your total time penalty gets decreased.

You can do at most K swaps. You can do the same swap multiple times if you want. Find out the minimum possible time penalty you can get if you are initially assigned problem S and you want to have problem D. You start with time penalty = 0. Note that you have to solve Q different queries as described in the input section.

Input Format

The first line will contain two integers, N (the number of problems) and M (the number of allowed swap operations). After that, M lines will follow. Each line will contain 3 integers u, v and w. This denotes that if you have problem u, you can swap it to get problem v with w time penalty. Then a single line will contain an integer Q, the number of queries. In each of the next Q lines, there will be three integers S, D and K. This means that for this query, S is the initial problem assigned to you, D is the problem you want and at most K swaps can be done.

Constraints

$1 <= N <= 50$

$1 <= M <= 500$

$1 <= u, v <= N$

$-100000 <= w <= 100000$

$1 <= Q <= 100000$

$1 <= S, D <= N$

$0 <= K <= 500$

There can be even swap operations where u and v are the same problems!

Output Format

For each of the queries, you have to print one line. The i-th line should contain the minimum possible time penalty for that query. If for a query, it is not possible to get the final problem, you should print "Impossible".

Sample Input 0

```
10 3
1 2 10
2 3 5
3 1 4
7
1 2 0
1 2 0
1 2 1
1 1 1
3 1 2
5 5 5
6 7 5
```

Sample Output 0

```
Impossible
Impossible
10
0
4
0
Impossible
```

Sample Input 1

```
2 2
1 2 4
2 1 -100
2
1 1 0
1 1 5
```

Sample Output 1

```
0
-192
```

By Association

Suppose that I know that $\alpha = a + b + c$ and $b = x + y + z$. I would like to conclude that $\alpha = a + x + y + z + c$. You would be surprised to learn that this does not follow by substitution!

In particular, $+$ is a left associative binary operator so what we really have is $\alpha = ((a + b) + c)$ and $b = ((x + y) + z)$, so that substituting yields: $\alpha = ((a + ((x + y) + z)) + c)$. Here the RHS is not definitionally the same as $a + x + y + z + c$, which is, if we recall, $((((a + x) + y) + z) + c)$.

You may protest, $+$ is associative, so no matter how you bracket an iterated sum, the results are all equal. However, associativity merely says that $((x + y) + z) = (x + (y + z))$ and it's not clear how this applies to more complicated expressions.

In this problem, you will devise an algorithm that **rebrackets** any expression to the standard left-associated form $((((a + b) + c) \ \ldots \ + z)$.

Input Format

The input will consist of a single fully bracketed expression. All indefinite terms (what we had as e.g. a) will be replaced with the letter X, and all operations will be suppressed.

For example, a valid input is given by `((X (X X)) (X X))`.

More formally, the set of valid inputs is defined by the grammar:

```
expr = X
     | (expr expr)
```

Constraints

The length of an expression is defined to be the number of `X`s that it contains.

In this problem the length of the input is ≤ 1000.

Output Format

The output will consist of some number of lines. Each line will consist of a **command** that performs one reassociation operation. A command consists of zero or more instances of the letter `L` or `R` followed by one of the letters `A` or `B`. More formally, the set of *syntactically well-formed commands* is defined by the grammar:

```
cmd = A
    | B
    | Lcmd
    | Rcmd
```

The interpretation of the commands is given as follows (we give both an informal description as well as a formal inference rule).

Here, the rules are given of the form: `cur-expr ⊢ cmd ⇒ next-expr`. The current expression is known as the *context* of a command.

The command `A` denotes the operation `((e f) g) ⇒ (e (f g))`.

$$\overline{((e\ f)\ g) \vdash \mathbf{A} \Rightarrow (e\ (f\ g))}$$

The command `B` denotes the operation `(e (f g)) ⇒ ((e f) g)`.

$$\overline{(e\ (f\ g)) \vdash \mathbf{B} \Rightarrow ((e\ f)\ g)}$$

Prefacing `L` to a command `α` applies `α` to the left subexpression of `(e f)`.

$$\frac{e \vdash \alpha \Rightarrow e'}{(e\ f) \vdash \mathbf{L}\alpha \Rightarrow (e'\ f)}$$

Prefacing `R` to a command `α` applies `α` to the right subexpression of `(e f)`.

$$\frac{f \vdash \alpha \Rightarrow f'}{(e\ f) \vdash \mathbf{R}\alpha \Rightarrow (e\ f')}$$

A command is **legal** in a given context if there exists a valid `next-expr` according to these rules. For example, in context `(X ((X X) X))` the command `B` is legal, while the command `A` is illegal. This is because the context is of the form `(e (f g))`, but not of the form `((e f) g)`.

When judging your solution, we will apply the first command of your output to the input expression, and every subsequent command to the resulting expression of the previous command (provided that everything is legal). Your ouptut will be judged correct if the expression resulting from applying all of your commands is in **left-associated form**.

Formally, the set of left associated forms can be defined by... you guessed it, a grammar:

```
lexp = X
     | (lexp X)
```

Note:

At a cartian point, if your output is vastly inneficient, then the checker will fail due to memmory leaks accumulating. We have significantly reduced the size of the legest test case in order to accomidate bloated output. If you see a `Custom Checker Failed` message on your output, then note that it is possible to make your output be at least 1000x smaller (measured in character length).

Sample Input:

```
((X (X X)) (X X))
```

Sample Output:

```
LB
B
```

Explanation:

The series of transformations induced by these two commands is:

```
((X (X X)) (X X))
⇒ (((X X) X) (X X))
⇒ ((((X X) X) X) X)
```

Note that any sequence of legal commands that transforms the input into left associated form will be accepted; there are multiple valid outputs for this input.

Sample Input 0

```
X
```

Sample Output 0

```
<The output should be totally empty, without any new-line characters!>
```

Sample Input 1

```
(X (X X))
```

Sample Output 1

```
B
```

Late Night Snacking

photo from pxfuel

Cathy is making cookies for the school bake sale and plans to start baking now. She is not perfect, so not all her cookies will be of the same size.

When a tray of cookies is done, she leaves it out to cool for 15 minutes before putting them away. For her last tray of cookies, she will leave it out and pack it in the morning as she will be tired. When she goes to bed, however, she knows that her son Max will take the largest cookie that has been left out. This is bad for Max's teeth especially before bed, so she wishes to give him the smallest cookie possible. She will choose the tray she will bake last in a way that will minimize the size of the cookie Max eats.

Can you help Cathy determine the size of the cookie Max will eat?

Input Format

The first line will contain the number of trays, T, followed by the number of cookies, C, in each tray.

The next T lines will contain C integers each, representing the sizes of the cookies, S, on each tray.

Constraints

- $1 \leq T \leq 100$
- $1 \leq C \leq 100$
- $1 \leq S \leq 100,000$

Output Format

Print the size of the cookie that Max will eat.

Sample Input 0

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

Sample Output 0

```
5
```

Explanation 0

The first tray has a maximum size cookie of 5. The second tray has a maximum size cookie of 7. Cathy will choose the first tray, so Max ends up eating a cookie of size 5.

Help Meow-t!

Toby the cat has decided to leave his home on earth to start a new, exciting life in the vast emptiness of space! He brought with him a box of 1 dimensional beams to create the framework for his new space home. When Toby finally opens his box, he notices that the beams, which are no longer moving, floated around the inside of the box in all sorts of ways, and now the beams are all scratched up! He'd like to remove the beams from the box one at a time without scratching them up any further. Toby decides the best strategy to do this is to choose a beam and carefully lift it in the positive Z direction until it is completely outside of the box. Once removed from the box, Toby can lay the beam out to the side and choose another beam to remove. A beam cannot be removed if lifting it out of the box would cause it to hit another beam. Keep in mind that the beam endpoints can also hit other beams and other beam endpoints!

Each beam is defined by two integer points corresponding to the endpoints of the beam. Beams can only be moved in the positive Z direction. Assume all beams have a non-zero length and initially no two beams intersect each other at any point.

Given the configuration of the beams, is there an order that the beams can be lifted out of the box without hitting any other beams?

Note: This is my cat Toby. He often sits like this pondering life's greatest questions. Here's one of Toby's classic quotes: "Meow meow meow meow meow" which roughly translates to "Better to have a short life that is full of what you like doing, than a long life, spent in a miserable way... it's absolutely stupid to spend your time doing things you don't like in order to go on doing things you don't like... it's so important to consider this question: What do I desire?"

Input Format

Input will be space separated on one line. The first integer is n - the number of beams in the box. The next $6n$ space separated integers are $x_1, y_1, z_1, x_2, y_2, z_2$ - the first and second endpoints of each beam.

Constraints

- $1 \leq n \leq 500$
- $-10,000 \leq x_1, y_1, z_1, x_2, y_2, z_2 \leq 10,000$

Output Format

Output "YES", if there is an order that the beams can be lifted from the box without hitting any other beams. Output "NO", if this is not possible.

Sample Input 0

```
2 -1 -1 0 1 1 0 -1 1 1 1 -1 1
```

Sample Output 0

```
YES
```

Explanation 0

Beam 2 can be removed then beam 1 can be removed.

Sample Input 1

```
3 -1 2 -1 2 -2 3 0 2 2 0 -1 -1 2 0 -1 -1 0 2
```

Sample Output 1

```
NO
```

Explanation 1

None of the beams can be lifted without hitting another beam.

Hacking ATMs

Eve has managed to hack into an ATM, but before she started to withdraw money, she noticed a trap the bank had left. The ATM only had access to specific digits, so any withdrawal of money that used digits that were not available would immediately be flagged. The exploit she found allows her to specify an amount to withdraw only once, but she can withdraw that specific amount repeatedly without being detected by the security system. Given the available digits, can you help her decide whether she can successfully withdraw a certain amount of money?

photo from Investopedia

Input Format

The first line contains two integers n — the amount of money that Eve seeks to withdraw, and k - the number of distinct digits that work for the ATM.

The second line contains the k space-separated distinct digits (0-9).

Constraints

- $1 \leq n \leq 10000$
- $1 \leq k \leq 4$

Output Format

If the number n can be withdrawn, print "YES". Otherwise, print "NO".

Sample Input 0

```
141 2
4 7
```

Sample Output 0

```
YES
```

Explanation 0

She can form 47, and withdraw 47 three times to get 141.

Sample Input 1

```
38 2
3 5
```

Sample Output 1

```
NO
```

Explanation 1

She can form 3, 5, 33, 35, 53, 55, etc..., but none of them allows Eve to withdraw 38 exactly. Note that she cannot mix and match (withdrawing 3 and 35 is not allowed).

Quads

Given N points in a plane, select 4 points that maximize the area of the quadrilateral with these four points as the vertices.

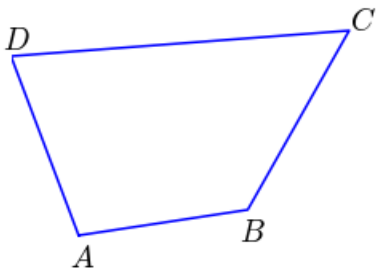


photo from Wikimedia

Input Format

The first line contains one integer N - The number of points.

The next N lines contain two space-separated integers x,y - representing points on a plane

Constraints

- $1 \leq N \leq 2 \cdot 10^3$
- $0 \leq x, y \leq 4 \cdot 10^4$

Output Format

A single containing a float - the maximum area possible from selecting 4 points.

Sample Input 0

```
50
20 33
53 12
81 28
72 31
65 13
40 10
15 74
80 37
22 31
39 24
7 98
54 14
17 50
14 41
71 94
74 80
49 19
17 44
18 63
94 15
39 47
60 23
```

```
96 36
94 2
26 39
86 93
73 65
7 69
74 83
10 9
42 63
27 36
30 94
46 100
64 2
3 76
99 19
11 1
33 70
10 74
59 91
16 93
83 78
3 99
1 44
36 52
22 29
52 71
2 84
56 96
```

Sample Output 0

```
7823.500
```