Problem A. Abnormal Words

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

"*Khoor!*" David exclaimed to Aram.

"*Jreeohghbjrrn!*" Aram responded.

Dismayed to find that club members weren't sufficiently confused, Aram and David decided to speak in a new language. Rather invent one from scratch, they decided to encode their speech with a Caesar cipher.

Specifically, they agree on the *Caesar shift s*. To say a word, they replace each of its letters by the letter that comes s places later in the alphabet. Letters that pass `z` wrap back around to `a`. For example if $s = 4$, `a` becomes `e`, `b` becomes `f`, and `y` becomes `c`.

Encoding and decoding words in their heads is very slow, so David and Aram asked you to write a program to automate this process. After all, they're still untangling their tongues after saying *jreeohghbjrrn*!

Input

The first line contains a single character, either "`E`" or "`D`", indicating whether Aram and David are requesting a word to encode or decode, respectively.

The second line contains a integer s ($1 \le s \le 25$), the shift.

The third line contains a single word w ($1 \le |w| \le 100$) consisting solely of the lowercase Latin letters from `a` to `z`.

Output

On a single line output the encrypted or decrypted word, as requested.

Examples

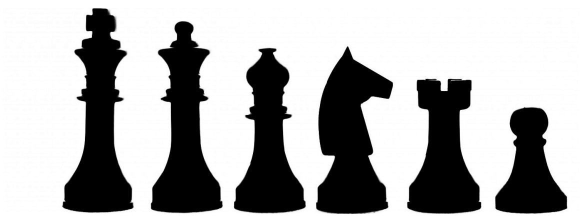
standard input	standard output

`E`
`3`
`hello`
	`khoor`

`D`
`3`
`jreeohghbjrrn`
	`gobbledeygook`

Problem B. Balanced Fighters

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes



Coaches Aram, Lucca, and David couldn't agree on who should order pizza. To avoid a fight in real life, they decided to battle in a video game while everyone else starves.

In this game, there are many fighters to choose from. Each fighter has a name and three integer *stats* HP, AT, DF: the fighters' starting health points, attack strength and defence armour. One-on-one combat between Fighters A and B progresses in a series of rounds. In each round, Fighter A's HP decreases by $\max(0, \mathrm{AT}_B - \mathrm{DF}_A)$, while Fighter B's HP *simultaneously* decreases by $\max(0, \mathrm{AT}_A - \mathrm{DF}_B)$. A fighter wins if, at the end of some round, their HP is positive while their opponent's HP is not. If this situation never occurs, the fight is ruled a draw with no winner.

Aram, Lucca, and David want a free-for-all. They decided that it would only be fun if they choose three fighters that form an *intransitive triple.* A, B and C are said to form an *intransitive triple* if, in one-on-one combat, A would win against B, B would win against C, and C would win against A.

Can you find all intransitive triples that Aram, Lucca, and David can choose from?

Input

On the first line is a single integer N ($1 \le N \le 100$), the number of fighters.

On each of the next N lines is a name consisting of 1 to 15 upper or lower case Latin letters, followed by three space-separated integers HP, AT and DF ($1 \le \mathrm{HP}, \mathrm{AT}, \mathrm{DF} \le 10\,000$) denoting the health points, attack strength, and defence armour for that fighter.

The names of all the fighters are distinct from one another.

Output

Output on the first line a single integer K, the number of intransitive triples.

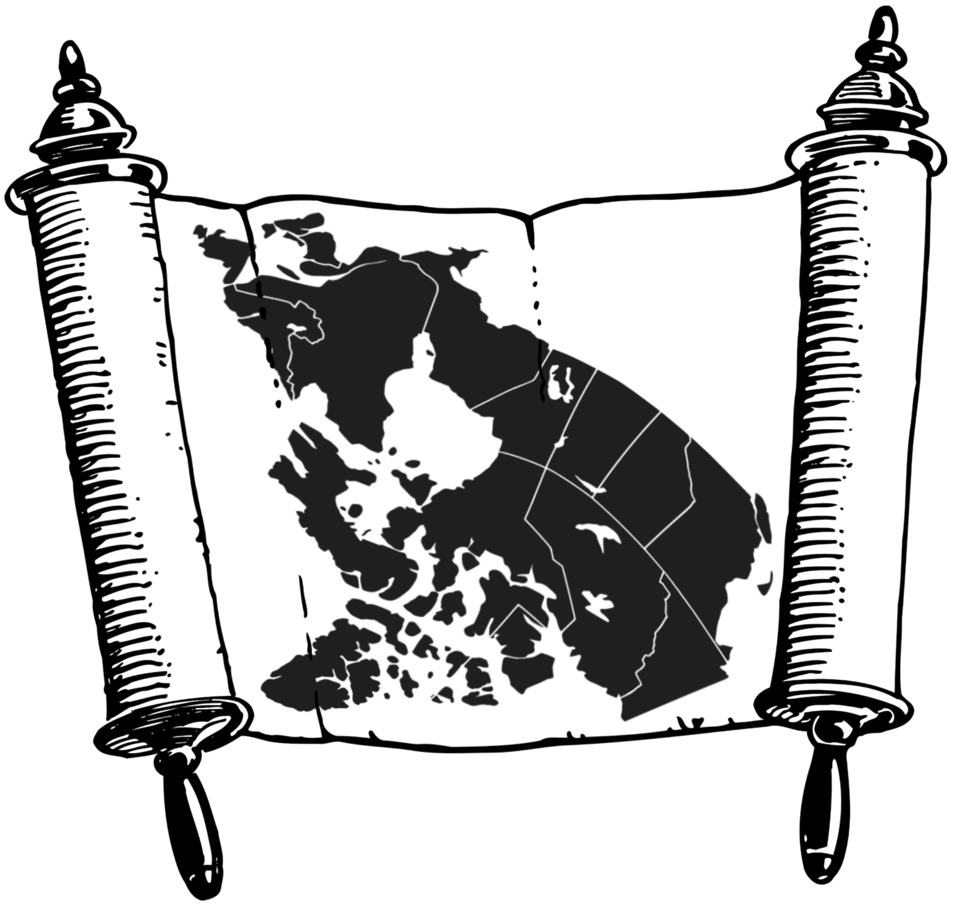
On each of the next K lines, output the names of three intransitive fighters separated by spaces. No two lines should describe the same set of fighters. Any ordering of intransitive triples, and of fighter names within an intransitive triple, will be accepted.

Examples

standard input	standard output
5 TheStrong 90 60 10 TheInvincible 10000 10000 10000 TheTough 70 50 25 TheBrick 3 1 4159 TheResilient 160 40 10	1 TheResilient TheStrong TheTough
1 TheLonely 500 500 500	0

Problem C. Unjob Search

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes



Aram got bored of having a job, so he quit and went off to the distant country of Lalaland.

Lalaland has N cities with unexciting names of $1, 2, \ldots, N$. There are $N - 1$ rail lines connecting pairs of cities, making it possible to get from any city to any other. In other words, Lalaland is a tree.

Lalaland has something called an *unjob*. Unjobs are similar to jobs, except that people who have an unjob can't be fired, have no obligations, and don't get paid. This is exactly what Aram is looking for so he can prove theorems all day long.

The Lalaland government knows that the general public views these unjobs as somewhat unsavory, and they may also be attracting large numbers of foreigners into the country. Hence they passed a law requiring people with unjobs to live in faraway *terminal cities*: cities that are only connected to one rail line. The government hoped that this would keep people with unjobs far away from the center of their country.

Aram wants to go to a terminal city. Having never been to Lalaland, he asked his cosmopolitan friend David if he knows of a terminal city. David said no, but claimed to have travelled between every pair of cities of Lalaland and can remember if any given city was on the unique shortest sequence of rail lines between them. He lets Aram ask up to N questions of the following form:

"For cities a, b, and c, do you pass by city b as you travel along the shortest path from a to c?"

If Aram asks more than N questions, David will get tired and take a nap for who knows how long. Help Aram quickly find a terminal city!

Interaction Protocol

The first line of input from the judge contains a single integer N ($3 \leq N \leq 2\,000$), the number of cities in Lalaland.

Following this, you may submit up to N queries of the following forms:

- `? a b c` — ($1 \leq a, b, c \leq N$) which asks if city b will be visited while taking the shortest set of rail lines from city a to city c.

The judge will respond with either 1 indicating that city b is indeed between cities a and c, or 0 indicating that it is not.

At any time, you may submit an answer in the following form:

- `! x` ($1 \leq x \leq N$) — which means your program claims that x is a terminal city.

After answering, your program must terminate immediately to get a verdict. Failing to follow this interaction protocol may result in unexpected errors.

If your input is malformed, the judge will output `-1`. Upon reading `-1` your program should exit immediately to get a wrong answer verdict, otherwise you may get another error.

Please be sure to use the stream flushing operation after each query in order not to leave part of your output in the buffer. For example, in C++ you can use `fflush(stdout)` or end with `cout << endl` or `cout << flush`. In Java you can use `System.out.flush()`. In Python, you can use `stdout.flush()`.

Example

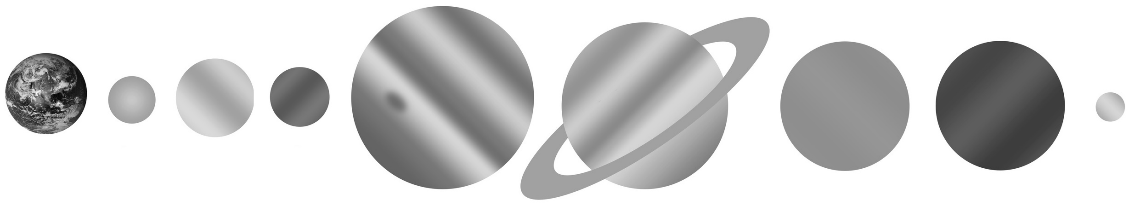
standard input	standard output
3	
	? 1 3 2
0	
	? 3 2 1
1	
	! 1

Note

The sample has two terminal cities, 1 and 3. Either answer would be accepted.

Problem D. Astrodirections

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes



It is a very rare event that all N colonized planets in the galaxy, numbered $1, 2, \ldots, N$, align in order on a straight line. To celebrate the occasion, a grand Astrofestival is held on one of the planets, where unhealthy amounts of Astrohol will be consumed.

Astrodavid doesn't know what Astrohol is, but he's very excited about the Astrogeometry meetup that will take place at the Astrofestival. Unfortunately, he doesn't know on which planet it's held. He may Astrojump between planets on his Astroship, but he has to buy fuel first. Luckily, he reached the fuel station on his home planet just before it closes for the Astrofestival. How much fuel must Astrodavid buy to guarantee that he'll make it to the Astrofestival?

Astrodavid lives on planet 1. His Astroship has jump power J. To take off from planet p, he chooses an Astrojump displacement i such $1 \le |i| \le J$, $1 \le p+i \le N$, and his fuel supply is at least f_i units. He will then land on planet $p+i$, his fuel supply having decreased by f_i units. Upon asking for directions there, he'll know if the Astrofestival is held at that planet, a different planet with a higher number, or one with a lower number. As long as he has enough fuel, he may choose to Astrojump again.

Input

The first line contains two space-separated integers N ($2 \le N \le 4\,000$) and J ($1 \le J \le \frac{N}{2}$), the number of colonized planets and the jump power, respectively.

The second line contains J integers, the i'th of which is f_i, the fuel cost of jumping upward by i planets.

The third line contains J integers, the i'th of which is f_{-i}, the fuel cost of jumping downward by i planets.

For all i, $0 \le f_i \le 10\,000$.

Output

Output the minimum number of units of fuel that Astrodavid must buy to guarantee that he'll reach the festival, no matter where it's actually held.

Examples

standard input	standard output

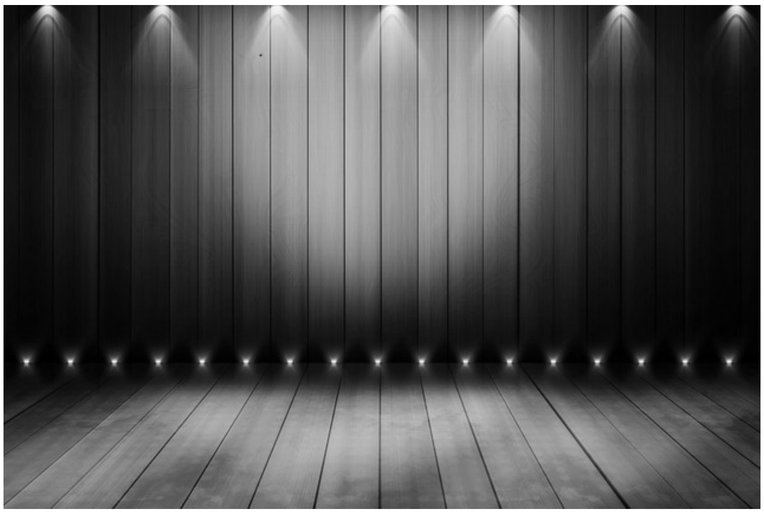
`16 8`
`5 5 5 5 5 5 5 5`
`5 5 5 5 5 5 5 5`
	`20`

`16 2`
`2 0`
`33 33`
	`30`

`10 5`
`50 60 70 80 90`
`5 4 3 2 1`
	`185`

Problem E. Exciting Acts

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes



Aram has decided to make an autobiographical play about his life. His script has N scenes with varying levels of excitement. For example, the scene of him studying for his first year exam was not very exciting, so Aram might rate it with an excitement level of 1. On the other hand, the scene with Aram at ICPC World Finals was very exciting, so its excitement level might be 1 000.

Aram needs to split the play into K acts. He knows that each act will only be remembered by the most exciting scene in it, so the excitement level of each act is the excitement level of the most exciting scene in the act. Since the scenes are chronologically ordered, each act must consist of a non-empty contiguous set of scenes. Every scene must appear in exactly one act.

The excitement of the play is the sum of excitements of the acts. Help Aram make the most exciting play!

Input

The first line contains two integers N and K ($1 \leq K \leq N \leq 2\,000$), the number of scenes and acts, respectively.

The second line contains N space-separated integers. The i'th integer is a_i ($1 \leq a_i \leq 1\,000$), the excitement level of the ith scene.

Output

Output a single integer, the maximum total excitement of the whole play.

Examples

standard input	standard output

`2 1`
`3 1`
	`3`

`2 2`
`3 100`
	`103`

Problem F. Fair Distribution

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

While wandering around the rural parts of Lalaland, Aram found himself in a backwards village so small that it wasn't named a number; instead, the village was named Lucca.

The N villagers of Lucca faced a dilemma. The government of Lalaland decided to build a dam that would flood all of Lucca, and paid the town a lump sum to compensate for the land and relocation costs. Following Lalaland tradition, the lump sum was set to the area of the union of all polygons whose vertices are a subset of the villagers' homes, also known as the area of the convex hull of the villagers' homes. Unfortunately, there were no clear instructions on how to distribute the money.

Yrneh, who lived on the edge of village, demanded a larger share because his house alone increased the size of the lump sum. Leinad, who lived near Yrneh, complained that even if Yrneh wasn't there, his location ensured that the lump sum would be about the same. Accul, who lived near the center of the village, demanded at least some fraction of the lump sum, at least for relocation since his home would also be destroyed.

Aram, appalled by the uncivilized nature of the discussion in the backwards village, decided to step in and help before it came to blows. He devised a brilliant solution. He would consider the villagers in a random order, given by a permutation p. Then, for each i, he would calculate A_i, the traditional lump sum using only the homes of $p_1, p_2, \ldots p_i$. For increasing the sum, the villager p_i would receive exactly their contribution $A_i - A_{i-1}$.

Yrneh complained that if he were unlucky enough to be considered first, he would get nothing. Aram thought for a bit and realized it would be more fair to take the expected value under this scheme, treating all $N!$ permutations as equally likely. He explained, using Lalaland's most sophisticated mathematics, that the total payout to the villagers would exactly equal the original lump sum.

This solution seems to have pleased everyone, so they turned to Aram to actually compute the amount of money each person would get. This is too much for Aram to compute by hand, so he needs your help!

Input

The first line contains N $(1 \le N \le 200)$, the number of villagers.

Then follow N lines, the ith of which contains two space separated integers x_i and y_i $(-10\,000 \le x_i, y_i \le 10\,000)$, indicating the planar coordinates of a point representing villager i's house. Homes are so small they can be treated as points. No two villagers have their homes at the exact same coordinates because all of them are single.

Output

Output N lines. On line i, output the amount of money villager i would get, as a floating-point number. If the exact answer is x, your answer y will be accepted if $\frac{|y-x|}{\max(x,1)} < 10^{-6}$.

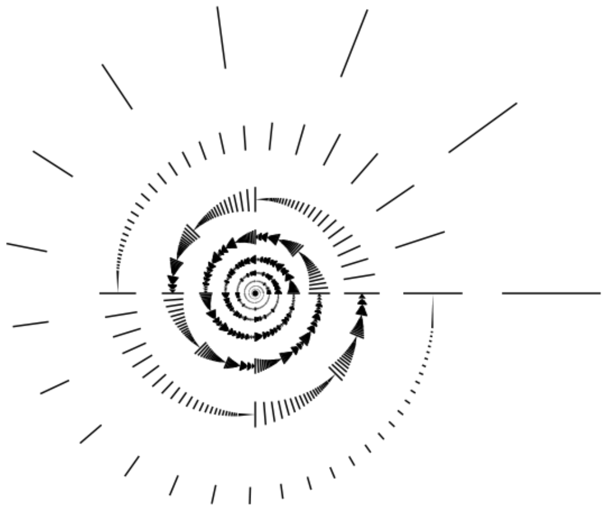
Example

standard input	standard output

`4`
`2 2`
`0 2`
`2 0`
`1 1`
	`0.8333333333333333`
	`0.5`
	`0.5`
	`0.1666666666666666`

Problem G. Infinity Plus One

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes



As children, David and Henry used to play "who can name the biggest number?"

Henry: "*Twenty!*"

David: "*Nine thousand!*"

Henry: "*A million trillion!*"

David: "*A googolplex squared!*"

Henry: "*Infinity!*"

Determined to win, David takes the game to the next level: "*Infinity plus one!*"

Lucca, passing by, objects: "*How can you count to infinity, let alone past it?*"

To give his winning move a mathematical basis, David invokes an ordered set $\mathcal{S}$. Only two properties of $\mathcal{S}$ are relevant:

- $\mathcal{S}$ is *well-ordered*: every non-empty subset of $\mathcal{S}$ has a unique smallest element.
- $\mathcal{S}$ is *uncountably big*: even if we take and name infinitely many elements from $\mathcal{S}$, there will always be more.

By taking one smallest element at a time, David identifies numbers with elements of $\mathcal{S}$:

- $0 = \min(\mathcal{S})$, the unique smallest element of $\mathcal{S}$,
- $1 = \min(\mathcal{S} \setminus \{0\})$, the smallest element in the subset that remains when 0 is removed from $\mathcal{S}$,
- $2 = \min(\mathcal{S} \setminus \{0, 1\})$, and so on...

By naming elements in this way, David can get all the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\} \subsetneq \mathcal{S}$.

In order to go beyond, he defines $\infty = \min(\mathcal{S} \setminus \mathbb{N})$. To get a "number" that's even bigger than ∞, he chooses $\min(\mathcal{S} \setminus (\mathbb{N} \cup \{\infty\}))$.

To make this idea general, David represents a "number" by a counting program P, which takes as input a (countable, possibly infinite) subset $T \subsetneq \mathcal{S}$. Its output, denoted by $\mathrm{P}(T)$, satisfies $T \subseteq \mathrm{P}(T) \subsetneq \mathcal{S}$. Counting programs belong to a context-free language:

- The empty program does nothing to T, returning it as-is.
- $+$ takes the smallest unused element of $\mathcal{S}$ and puts it in T. Formally, $+(T) = T \cup \{\min(\mathcal{S} \setminus T)\}$.
- PQ runs the program P, followed by the program Q. Formally, $(\mathrm{PQ})(T) = Q(\mathrm{P}(T))$.
- [P] runs the program P in an "infinite loop". Formally, $[\mathrm{P}](T) = \mathrm{P}(T) \cup (\mathrm{PP})(T) \cup (\mathrm{PPP})(T) \cup \ldots$

Let $T_1 = \mathrm{P}_1(\emptyset)$ and $T_2 = \mathrm{P}_2(\emptyset)$ be the subsets constructed by the counting programs P_1 and P_2, respectively. The following conditions are equivalent, and when they hold we'll say that $\mathrm{P}_1 < \mathrm{P}_2$:

- $\min(\mathcal{S} \setminus T_1) < \min(\mathcal{S} \setminus T_2)$
- $T_1 \subsetneq T_2$

Thanks to his general definition, David can be mathematically precise about what he means by "infinity plus one": he means the program $[+]+$, which constructs the subset $([+]+)(\emptyset) = \mathbb{N} \cup \{\infty\}$.

To decide who won their game, the children need you to compare numbers that are described by counting programs. Given a list of M counting programs, sort them from smallest to biggest.

Input

The first line contains an integer M ($M \le 100$).

Line $i+1$ contains a string P_i ($1 \le |\mathrm{P}_i| \le 100$). P_i is guaranteed to be a counting program in the described language: all characters are `+`, `[`, or `]`, and every `[` is matched with a corresponding `]`.

Output

Print the list of counting programs in sorted order, from smallest to biggest. Counting programs that are equal may be printed in any order.

Example

standard input	standard output
8 + [+]+ [+][+] +++++++ +++[+++] +[+[+]+]+ [+][[+]][+] [][[][]][]	[][[][]][] + +++++++ +++[+++] [+]+ [+][+] +[+[+]+]+ [+][[+]][+]

Note

The symbols $\subseteq$, $\subsetneq$, $\cup$, $\setminus$, $\emptyset$ mean "is equal to or contained in", "is strictly contained in", set union, set difference, and the empty set, respectively.

The sample numbers are all distinct; study them carefully using the definitions. Intuitively, you might think of the smallest entries as zero, one, seven, infinity, infinity plus one, and infinity times two, respectively. But don't let names fool you: the usual rules of arithmetic no longer apply!

Problem H. Ancient Wisdom

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

$$CD^3 = A^2$$

David and Aram had the following conversation:

David: "*Aram, you're pretty old.*"

Aram: "*I'm not THAT old. If you cube your age and then multiply by C, you'd get exactly my age squared.*"

Lucca overheard this conversation but isn't sure if he caught the right value for C. Help Lucca figure out how young David could be, assuming he heard correctly.

As everyone knows, ages are positive integers.

Input

Input is a single integer, what Lucca thought he heard for the value of C ($1 \leq C < 2^{63}$).

Output

Output the minimum age David could be.

Examples

standard input	standard output
1029	21
10	10
1000000000000000000	1

Note

The average computer can do roughly 10^8 elementary operations in one second. If you submit a program that tries to do more than 10^9 operations, you shouldn't be surprised if your program gets a Time Limit Exceeded verdict!