Problem P
Turning Red
Time limit: 3 seconds

Mei's parents have spent the last year remodeling their house, but their lighting system is quite complex! Each room in the house has an LED light, which can be set to red, green, or blue, as seen in Figure P.1.

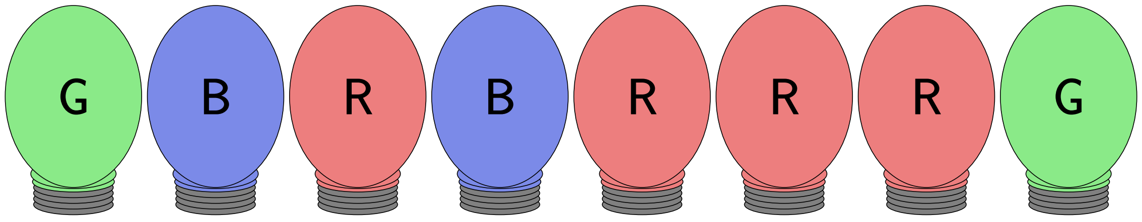

Figure P.1: The initial state of the lights in Sample Input 1. Buttons and wires not shown.

Throughout the house are various buttons which are each connected to one or more lights. When a button is pressed, any red lights connected to that button become green, any green lights connected to that button become blue, and any blue lights connected to that button become red. Each button can be pressed multiple times. Because the house was built prior to the invention of crossbar wiring, each light is controlled by at most two buttons.

Mei's favorite color is red, so she wants to turn all of the lights red. Her parents, fearing the buttons will wear out, have asked her to minimize the total number of button presses.

Input

The first line of input contains two positive integers l and b, where l $(1 \le l \le 2 \cdot 10^5)$ is the number of lights and b $(0 \le b \le 2 \cdot l)$ is the number of buttons. The second line of input is a string of l characters, all either `R`, `G`, or `B`, where the i^{th} character is the initial color of the i^{th} light. The next b lines describe the buttons. Each of these lines begins with an integer k $(1 \le k \le l)$, the number of lights controlled by this button. Then k distinct integers follow, the lights controlled by this button. The lights are indexed from 1 to l, inclusive. Each light appears at most twice across all buttons.

Output

Output the minimum number of button presses Mei needs to turn all the lights red. If it is impossible for Mei to turn all of the lights red, output `impossible`.

Sample Input 1

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

Sample Output 1

```
8
```

Sample Input 2

```
4 3
RGBR
2 1 2
2 2 3
2 3 4
```

Sample Output 2

```
impossible
```

Sample Input 3

```
4 4
GBRG
2 1 2
2 2 3
2 3 4
1 4
```

Sample Output 3

```
6
```

Sample Input 4

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

Sample Output 4

```
3
```

Problem Q
Doing the Container Shuffle
Time limit: 2 seconds

Cargo containers by Martini171 via Wikimedia Commons, cc by-sa

Majestic cargo ships, each carrying thousands of shipping containers, roam the world's seas every day. They make modern trade possible by being so efficient that shipping goods halfway around the world costs only pennies.

Once the ships reach their destination, their standard-size cargo containers are unloaded from the ship onto stacks on land, from which they are moved to trains or trucks that deliver them to their destination. It turns out that moving containers is expensive, so port operators try to minimize the number of moves necessary for delivering cargo.

In this problem, we consider such a container-unloading scenario. We need to unload n containers, which are placed into two stacks built from bottom to top. The placement of each container is at random, with equal probability it will be put onto the first or the second stack (independently of other containers). Once all containers are unloaded, they will be picked up by trucks in a given order. When a truck wants to load a specific container, there are two cases. If the container is on top of its stack, then the container can be moved to the truck without moving any other containers. Otherwise, containers have to be moved from one stack to the other until the requested container is at the top of its stack. At that point the container can be moved onto the truck.

As an example, consider a case of three containers that arrive in order `1`, `2`, `3`. Assume that `1` and `3` are in the first stack, and `2` is in the second. If the containers are moved onto trucks in order `1`, `2`, `3`, then five moves of containers have to take place:

Stack 1	Stack 2	Comment
`1 3`	`2`	Initial configuration (stacks bottom to top)
`1`	`2 3`	Move container `3` from stack 1 to stack 2
	`2 3`	Move container `1` to truck
`3`	`2`	Move container `3` from stack 2 to stack 1
`3`		Move container `2` to truck
		Move container `3` to truck

Table Q.1: Example moves of containers requested in order `1 2 3`.

We want to know how many moves are necessary to deliver all containers to the customers. Assuming that container placement is random, we ask you to compute the expected number of moves necessary for a given truck-loading order.

Input

The first line of input contains an integer n ($1 \leq n \leq 10^6$), the number of containers. The containers are numbered $1, 2, \ldots, n$, and are unloaded from the ship in this order. The second line of input contains n integers $a_1, \ldots, a_n$. These numbers are a permutation of $\{1, 2, \ldots, n\}$, and specify the order in which the containers are loaded onto trucks.

Output

Output the expected number of moves necessary to load the containers onto the trucks — this excludes the cost of unloading them from the ship, but includes both moves between stacks and from a stack to a truck. Your answer should have an absolute error of at most 10^{-3}.

Sample Input 1

```
5
4 2 5 3 1
```

Sample Output 1

```
7.000
```

Sample Input 2

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

Sample Output 2

```
13.500
```

Problem R
Zoo Management
Time limit: 5 seconds

When managing a zoo, you sometimes want to move the animals between enclosures. You might figure out that the zebras will enjoy the spacier enclosure currently occupied by the penguins, while the penguins might want to move to the colder enclosure where the koalas currently live; and koalas will move to an empty enclosure that can be filled up with eucalyptus. So, you would move the koalas first, to free up the colder enclosure, then move the penguins there, and finally move the zebras.

A picture of a zoo with a camel, a donkey and (part of) a goat.

The way you move animals is by using special tunnels that connect the enclosures — you do not want the animals to move outside, both because of the risk that they will be scared, and because of the risk that they might run away and hurt themselves.

Unfortunately, you have acquired more animals recently, and all the enclosures are now full, which makes moving animals around much harder. Imagine, for instance, that the koalas were to move to the former zebra enclosure — you cannot move any set of animals first. Instead, what you learned to do, is to move the animals at the same time — the zebras, koalas and penguins start moving down different tunnels at the same time, and arrive at their new enclosures at the same time — and thus they never meet. Note that you cannot just swap the animals in two connected enclosures this way (because they would meet in the tunnel and become scared).

So, now you have a puzzle. You have a list of enclosures, each with an animal type inside; some of those enclosures are connected by tunnels. You can, any number of times, choose some set of animals and move each one to an enclosure adjacent by tunnel, as long as the animal in that enclosure is also moved as the part of the same move, and no tunnel is used more than once as a part of the same move. You also have your vision of the perfect way to position the animals. Is it possible to do so, in a series of moves?

Input

The first line of the input consists of two integers n ($1 \le n \le 4 \cdot 10^5$) and m ($0 \le m \le 4 \cdot 10^5$), indicating the number of enclosures and tunnels. Then follow n lines, the i^{th} of which contains two integers b_i ($1 \le b_i \le 10^6$) and e_i ($1 \le e \le 10^6$), indicating the type of animal that is in enclosure i at the beginning, and the type of animal that you want to be in enclosure i after all the moves. You may assume that $e_1, \ldots, e_n$ is a permutation of $b_1, \ldots, b_n$.

Then follow m lines describing the tunnels. Each line contains two integers x and y ($1 \le x < y \le n$), indicating that enclosures x and y are connected by a two-way tunnel. No two enclosures are connected by more than one tunnel.

Output

If it is possible to move the animals so that every enclosure contains the desired animal type, output `possible`. Otherwise, output `impossible`.

Sample Input 1

```
3 3
1 4
4 7
7 1
1 2
2 3
1 3
```

Sample Output 1

```
possible
```

Sample Input 2

```
2 1
1 2
2 1
1 2
```

Sample Output 2

```
impossible
```

Sample Input 3

```
5 6
10 40
20 30
30 50
40 20
50 10
1 2
2 3
1 3
3 4
3 5
4 5
```

Sample Output 3

```
possible
```

Sample Input 4

```
4 4
10 10
10 20
20 10
20 20
1 2
2 3
3 4
1 4
```

Sample Output 4

```
impossible
```

Problem S
Bridging the Gap
Time limit: 4 seconds

A bridge with low capacity

A group of walkers arrives at a river in the night. They want to cross a bridge, which can hold a limited number of walkers at a time. The walkers have just one torch, which needs to be used when crossing the bridge. Each walker takes a certain time to cross; a group crossing together must walk at the slowest walker's pace. What is the shortest time it takes for all walkers to cross the bridge?

For example, Sample Input 1 assumes the bridge can hold 2 walkers at a time and there are 4 walkers with crossing times 1 minute, 2 minutes, 5 minutes and 10 minutes, respectively. The shortest time of 17 minutes can be achieved by the following sequence of crossings. First, the two fastest walkers cross in 2 minutes. Second, the fastest walker crosses back in 1 minute. Third, the two slowest walkers cross in 10 minutes. Fourth, the second-fastest walker crosses back in 2 minutes. Fifth, the two fastest walkers cross in 2 minutes.

Input

The first line of input contains two integers n and c, where n ($2 \leq n \leq 10^4$) is the number of walkers, and c ($2 \leq c \leq 10^4$) is the number of walkers the bridge can hold at a time.

Then follows a line containing n integers $t_1, \ldots, t_n$ ($1 \leq t_i \leq 10^9$ for all i). The i^{th} walker takes time t_i to cross.

Output

Output the minimum total time it takes for the entire group to cross the bridge.

Sample Input 1	Sample Output 1
4 2 1 2 10 5	17

Sample Input 2	Sample Output 2
4 6 1 2 10 5	10

Problem T
Carl's Vacation
Time limit: 1 second

Carl the ant is back! After traversing meandering paths (Problem A, 2004 World Finals) and wandering over octahedrons (Problem C, 2009 World Finals) it is time for a little vacation — time to see the sights! And where better to see the sights than at the tips of tall structures like, say, pyramids!! And where better to see tall pyramids but Egypt!!! (This is so exciting!!!!!)

After taking in the view from the tip of one pyramid, Carl would like to go to the tip of another. Since ants do not do particularly well in the hot sun, he wants to find the minimum distance to travel between the tips of these two pyramids, assuming he can only walk on the surfaces of the pyramids and the plane which the pyramids sit upon. The pyramids are, geometrically, right square pyramids, meaning the apex of the pyramid lies directly above the center of a square base.

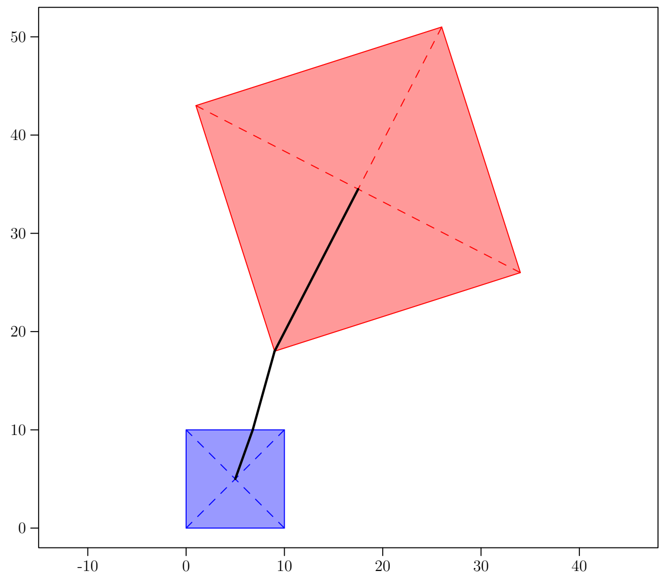

Figure T.1: Illustration of two pyramids corresponding to Sample Input 1. The black line shows the shortest path between their apexes.

Input

The first line of input contains five integers x_1, y_1, x_2, y_2, h where x_1, y_1, x_2, y_2 ($-10^5 \le x_1, x_2, y_1, y_2 \le 10^5$ and $(x_1, y_1) \ne (x_2, y_2)$) define an edge of the first pyramid, with the body of the pyramid lying to the left of the directed vector from (x_1, y_1) to (x_2, y_2), and h ($1 \le h \le 10^5$) is the height of the pyramid. The second line of input describes the second pyramid in the same format. The intersection of the bases of the two pyramids has 0 area.

Output

Output the minimum distance Carl travels between the tips of the two pyramids. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1

`0 0 10 0 4`
`9 18 34 26 42`
	`60.866649532`

Problem U
Toy Train Tracks
Time limit: 1 second

Model railroad by Marshman via Wikimedia Commons, cc by-sa

Every little child, and quite a number of adults, are fascinated by toy trains. From a toddler's choo-choo train to a hobbyist's elaborate model railroad filling an entire basement, they are a profitable business. The Toy Train Tracks Construction Company (TTTCC) manufactures train tracks for all ages and skill levels. To keep their existing customers busy and maybe attract some new ones, the TTTCC has recently started publishing maps for how to connect their train tracks into elaborate layouts. Usually, this starts with a designer coming up with an interesting track layout, and then publishing both the layout and the required number of different track segments (say, curves and straight parts) needed to construct it. But the TTTCC has recently learned that many customers are looking for the reverse: they already have train track segments lying around (maybe found in grandma's attic), and would like to use them to create a large train course. How difficult might that be?

To study the feasibility of automating the layout-creation process, TTTCC is interested in constructing train courses using two different shapes: straight line segments, and 90-degree turns (see Figure U.1).

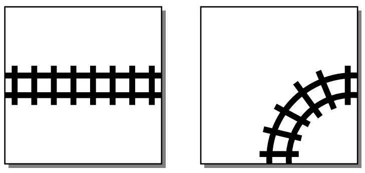

Figure U.1: A straight track segment and a curved track segment.

Valid layouts are created by placing these shapes on a square grid, with each track piece taking up exactly one grid cell. Both types of pieces can be rotated in 90-degree increments. A "proper" train track needs to be connected, and should form a single closed loop. Given a set of straight and curved track segments, what is the longest closed loop that one can construct?

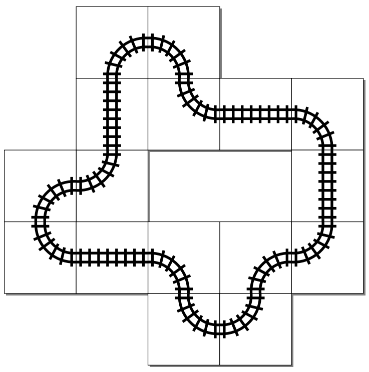

Figure U.2: A sample track using four straight track segments and twelve curves. This corresponds to Sample Output 1.

Input

The input consists of a single line containing two integers s and c, the number of straight and curved track segments available, respectively ($0 \leq s \leq 10^5$, $4 \leq c \leq 10^5$).

Output

Output a train loop using at most s straight segments and c curved segments, that has the longest length (in number of track segments used) under this restriction. The loop must be closed and cannot intersect itself. If there are multiple loops of maximal length, any one of them will be accepted.

If the loop is of length n, then print a single string of length n, where the characters represent the loop's segments as encountered in a single traversal. The character '`S`' stands for a straight-line segment, '`L`' for a curved segment that is a left turn, and '`R`' for a curved segment that is a right turn.

Sample Input 1	Sample Output 1
`4 12`	`LSRLLRLSLSRLLSRL`

Sample Input 2	Sample Output 2
`1 5`	`LLLL`

Problem V
Three Kinds of Dice
Time limit: 1 second

See how they roll! According to a famous story, Warren Buffett once challenged Bill Gates to a simple game of dice. He had three dice; the first player could examine them and choose one of the three. The second player would then choose one of the remaining dice, and both players would roll their dice against each other, aiming for the highest numbers. Warren offered to let Bill go first, but this made Bill suspicious so he opted to go second. It turned out to be a wise choice: these were *intransitive dice*. The first die had an advantage when rolling against the second, the second had an advantage when rolling against the third, but the first did not have an advantage when rolling against the third!

Image from rawpixel, CC0

To formalize this: define a "die" as any shape with at least one face such that each face shows a positive integer. When a die is rolled, one of its faces is selected uniformly at random. When two dice roll against each other, the die whose selected face shows a higher number earns 1 point; if both numbers are equal, each die earns $\frac{1}{2}$ points. For dice D and D', define $score(D, D')$ as the expected number of points D earns from a single roll against D'. If $score(D, D') > \frac{1}{2}$, we say that D has an advantage over D'; if $score(D, D') = \frac{1}{2}$, the two dice are tied. For example, if D is the first die in the sample input and D' is the second, $score(D, D') = \frac{4}{9}$ and $score(D', D) = \frac{5}{9}$, so D' has an advantage over D.

Given two dice D_1 and D_2 such that D_1 has an advantage over D_2, you want a third die D_3 that forms an intransitive trio with the other two. Among all D_3 that have an advantage over or tie with D_1, compute the lowest possible $score(D_3, D_2)$. If this is less than $\frac{1}{2}$, you can make an intransitive trio! Similarly, among all D_3 such that D_2 has an advantage over or ties with D_3, compute the highest possible $score(D_3, D_1)$.

Input

The input contains two lines, each describing one die. One of the dice (the first or the second) has an advantage over the other. The die with the advantage is D_1 and the other is D_2.

The first integer on a line gives n ($1 \leq n \leq 10^5$), the number of faces on the die. Then follow n integers f_i ($1 \leq f_i \leq 10^9$ for each $1 \leq i \leq n$), giving the integer on each face.

Output

Output one line containing the lowest $score(D_3, D_2)$ and the highest $score(D_3, D_1)$ under the above conditions. The two scores do not need to use the same die D_3. Your answer should have an absolute error of at most 10^{-6}.

Sample Input 1	Sample Output 1

`6 1 1 6 6 8 8`
`3 2 4 9`
	`0.291666667 0.750000000`

Sample Input 2

```
4 9 3 7 5
3 4 2 3
```

Sample Output 2

```
0.500000000 0.500000000
```

Problem W
Riddle of the Sphinx
Time limit: 2 seconds

One of the most remarkable landmarks in Egypt is the Great Sphinx of Giza, a statue depicting a mythical creature with the head of a human, the body of a lion, and the wings of an eagle. Sphinxes were regarded as guardians in Egyptian and Greek mythologies. Probably the most famous sphinx is the one who guarded the Greek city of Thebes. According to myths, when Oedipus tried to enter the city, the sphinx gave him the following riddle: "Which creature has one voice, but has four feet in the morning, two feet in the afternoon, and three feet at night?" As you might have heard, Oedipus correctly answered, "Man — who crawls on all fours as a baby, then walks on two feet as an adult, and then uses a walking stick in old age."

Oedipus and the Sphinx by Gustave Moreau, 1864, public domain

In this problem, you meet a different sphinx who gives you a somewhat reversed riddle: "How many legs do an axex, a basilisk, and a centaur have?" While you recognize these as creatures from Egyptian and Greek mythology, you have no clue how many legs each has (except that it is a nonnegative integer). The sphinx sternly instructs you to not touch anything so you are unable to search for the answer on your phone.

However, the sphinx allows you to ask her five questions. In each question you can ask the sphinx how many legs some number of these creatures have in total. For instance, you could ask, "How many legs do three basilisks and one axex have in total?" or "How many legs do five centaurs have?" Seems easy enough, you think, but then you remember that sphinxes are tricky creatures: one of the sphinx's five answers might be an outright lie, and you do not know which one.

Write a program to talk to the sphinx, ask the five questions, and solve the riddle.

Interaction

There are exactly five rounds of questions. In each question round, you must first write a line containing three space-separated integers a, b, and c ($0 \leq a, b, c \leq 10$), representing the question "How many legs do a axex, b basilisks, and c centaurs have in total?" After the question is asked, an input line containing a single integer r ($0 \leq r \leq 10^5$) is available on standard input, giving the sphinx's answer to your question.

After the five rounds of questions, output a line containing three space-separated nonnegative integers ℓ_a, ℓ_b, and ℓ_c, indicating the number of legs of an axex, a basilisk, and a centaur, respectively.

Sample Interaction 1

Read	Write
	1 1 1
12	
	1 1 1
13	
	5 0 1
24	
	1 0 0
4	
	1 1 0
8	
	4 4 4

Sample Interaction 2

Read	Write
	4 4 4
2023	
	1 0 0
0	
	0 1 0
42	
	0 0 1
2024	
	0 0 0
0	
	0 42 2024

Problem X
Quartets
Time limit: 2 seconds

1
Franz
Baſen, Vetter, Butter
Die hübſchen Bäschen bitten ſehr:
„Ach, Vetter Franz, reit' mal umher!"
Und Franz, natürlich gleich bereit,
Gewinnt das Tier durch Freundlichkeit.
2
Baſen
Franz, Vetter, Butter
Schon ſitzt er d'rauf und kommt nicht weiter,
Worob die Baſen äußerſt heiter.
Er denkt: „Ja wart'! du wirſt ſchon geh'n!
Ich muß dich 'mal beim Schwanze dreh'n!"
3
Vetter
Franz, Baſen, Butter
Und er, der eben noch verſtockt,
Ermuntert ſich und ſpringt und bockt.
Im Eck, wo die Geräte hängen,
Sucht er den Vetter zu be-drängen.
4
Butter
Franz, Baſen, Vetter
Und — ſchwupp! — kommt Vetter Franz im Bogen
Auf die Couſinen zugeflogen.
Und — pautz! perdautz! — geht's über Kopf
Durch Butter und durch Millichtopf.
Four cards of a set, from a German Quartets deck by Wilhelm Busch via Wikimedia Commons, public domain

You are watching a group of kids playing a friendly game of Quartets and having fun. At one moment, you started being suspicious that some of the kids may be cheating. It seems the kids themselves are not bothered by that at all. Quite the contrary, this brings them even more fun, especially when someone gets caught cheating. As a programmer, you immediately started to think about ways to detect cheating by just observing their game.

Quartets is a 4-player card game that uses a deck of 32 cards divided into 8 sets of 4 cards each. Actual cards often contain educational images, which helps players to learn not only individual objects but also their classification. For example, the cards may display animals while the sets correspond to various groups of animals (mammals, reptiles, etc.).

At the beginning of a Quartets game, every player is dealt 8 cards and player 1 starts their turn. In each turn, a player asks another player whether they have one particular card. If the asked player has that card, they must give it to the requester, and the requester continues their turn by asking any of the players for another card. If the asked player does not possess the requested card, the requesting player loses the turn and the asked player starts their turn. An important rule states that the requesting player must already have at least one card from the same set as the card they ask for.

If, during their turn, a player holds a full set ("quartet") in their hand, the player may show the quartet to the other players, put it aside and gain one point. The four cards of the quartet are permanently removed from the current game. If at any point a player has no cards left, that player leaves the game. If it was that player's turn, the turn passes to the next player in sequence (player 1-2-3-4-1-...) who still has any cards. If no players have any cards, the game ends and the player with the most points is declared the winner.

Players are not allowed to ask for cards that have been removed from the game, and they cannot ask for a card from a player who has left the game.

Cheating in the game of Quartets is conceivable in two situations where a player falsely claims to have (or not have) some card. Specifically, the cheating occurs if

- a player asks for a card although they have no card of the corresponding set, or
- a player being asked claims not to have the requested card although they have it.

Needless to say, cheating is often discovered later. Theoretically, a watchful opponent can eventually find out about any cheating, as all cards are finally revealed at some point before the game ends.

Note that asking for a card that the asked player cannot possibly have (for example, because the requester has it in their own hand) is not exactly a smart move but is not considered cheating by itself.

Input

The first line of input contains an integer n ($1 \leq n \leq 1000$), the number of consecutive actions in one game of Quartets, starting from the beginning of a game. The next n lines each contain a description of an action. Each action is described by one line and may be one of the following:

- x `A` y sk `yes` — player x asks player y for the card sk, and player y hands that card to player x;
- x `A` y sk `no` — player x asks player y for the card sk, player y claims not to have it, and starts their turn;
- x `Q` s — player x puts aside a quartet s.

In all actions, x and y ($1 \leq x, y \leq 4$, $x \neq y$) are integers denoting the players, s ($1 \leq s \leq 8$) is a one-digit integer corresponding to one of the sets/quartets, and $k \in \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$ is a character identifying the four cards within a set. Thus the cards sk are labeled `1A`, `1B`, `1C`, `1D`, `2A`, `2B`, $\ldots$, `8C`, `8D`. The four cards that share the same first digit form a single set.

The given sequence describes a valid game being played among 4 players. That is, except for the two possible ways of cheating mentioned above, all actions satisfy all the rules.

Output

If there is a possible initial distribution of the cards such that the sequence of actions corresponds to a (possibly partial) game with all players following the rules, output `yes`. Otherwise, output `no`, followed by a line giving the number of the first action after which it is certain that some player must have cheated. Actions are numbered sequentially starting with 1, the actions of putting quartets aside are also counted.

Explanation of Sample Input 1

Players 1 and 2 both claim not having `3C`, and neither of them had `3A` or `3D`. So one of them either lied about not having `3C`, or asked for `3C` without having `3B` (the only remaining card from set 3).

Sample Input 1

```
4
1 A 2 3C no
2 A 3 3A yes
2 A 4 3D yes
2 A 1 3C no
```

Sample Output 1

```
no
4
```

Sample Input 2

```
6
1 A 2 3C no
2 A 3 3A yes
2 A 4 3D yes
2 A 1 3B no
1 A 4 5B yes
1 Q 5
```

Sample Output 2

```
yes
```

Problem Y
Compression
Time limit: 1 second

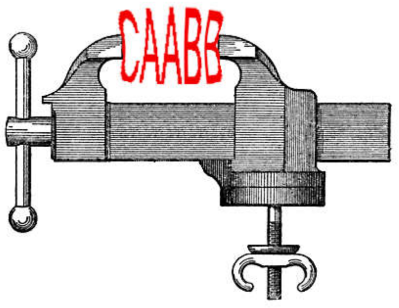

Image adapted from Wikimedia Commons, public domain

Infinite Compression Plan Consortium (ICPC) has developed a new, great data compression strategy, called "Don't Repeat Yourself" (DRY). DRY works on a string of characters, and if the string contains two consecutive instances of the same substring, it simply removes one of them. For example, in the string "`alfalfa seeds`", it could remove one of the two "`e`" substrings in "`seeds`", and one of the two "`lfa`" substrings in "`alfalfa`", resulting in "`alfa seds`". DRY can also take advantage of previous removals — for instance, in the string "`seventeenth baggage`", it will first remove the duplicate "`e`" in "`seventeenth`" and the duplicate "`g`" in "`baggage`", resulting in "`sevententh bagage`", and then remove the duplicate "`ent`" in "`sevententh`" and "`ag`" in "`bagage`", resulting in "`seventh bage`".

If there are multiple choices of repeating consecutive substrings to remove, DRY should choose in a way that results in the shortest possible final string. For example, in the string "`ABBCDCABCDCD`", DRY has two choices — either removing the repeated "`B`" near the beginning, or the repeated "`CD`" at the end. If the "`B`" is removed, then DRY will be able to remove the repeated "`ABCDC`", resulting in "`ABCDCD`", from which the "`CD`" at the end will finally be removed, resulting in "`ABCD`". However, if DRY removed the "`CD`" at the end first, it would be left with "`ABBCDCABCD`", from which only the repeated "`B`" can be removed to obtain "`ABCDCABCD`" — and from that string nothing more can be removed. So, the correct choice for DRY is to begin by compressing the double "`B`", and to finally end up with "`ABCD`".

ICPC observed that DRY obtains the best results on binary strings — that is, strings composed only of zeroes and ones. So, it has tasked you with implementing the best possible DRY algorithm working on binary strings. For any binary string, you should output a shortest string that can be obtained by repeatedly applying DRY to it.

Input

The input consists of a single line containing a nonempty string of length less than or equal to 10^5, consisting only of zeroes and ones.

Output

Output one line, containing a shortest possible result of running DRY on the input string. If there is more than one possible shortest result, any one will be accepted.

Sample Input 1	Sample Output 1
`1111`	`1`

Sample Input 2	Sample Output 2
`101`	`101`

Sample Input 3	Sample Output 3
`10110`	`10`

Problem Z
Archaeological Recovery
Time limit: 1 second

Renowned professor of Egyptology Z Mummer is exploring a newly discovered tomb in Luxor where she finds a mysterious construction. On a wall, there is a row of k pyramid-shaped stone slabs. Each stone features three hieroglyphs: an ankh (top), an eye of Horus (bottom left), and an ibis (bottom right).

Next to the wall there are n levers. Cautiously experimenting with these, wary of potential traps, the professor realizes that each lever rotates some of the pyramids clockwise or counter-clockwise so that another hieroglyph is pointing upwards on them. Flipping back a lever rotates the same pyramids in the opposite direction back to their original position. The levers operate completely independently of each other, so flipping or unflipping a lever has the same effect regardless of what states the other levers are in. See Figure Z.1 for an illustration.

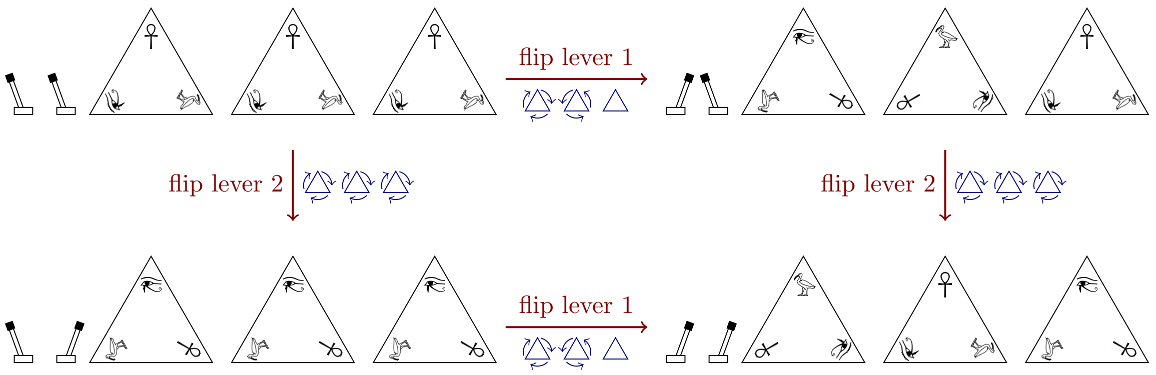

Figure Z.1: Illustration of a wall with $k = 3$ pyramids and $n = 2$ levers. The first lever rotates the first pyramid clockwise, the second pyramid counterclockwise, and leaves the third pyramid unchanged. The second lever rotates all three pyramids clockwise. This corresponds to Sample Input 1.

Intrigued, Professor Mummer records the individual effect of each lever. Back at her university after the expedition, she assigns one of her students the laborious task of figuring out all 2^n possible pyramid configurations (some of which may be identical) that can be achieved by flipping some subset of the levers, so that these can be studied further.

After many nights of careful calculations, the student is finally done and starts gathering his papers. But then disaster strikes: he accidentally spills some ink and completely destroys the professor's original notes, which contained the only record of the individual effects of each lever.

The only chance to escape Professor Mummer's wrath is to reconstruct the original notes from the list of possible pyramid configurations. This cannot be done completely unambiguously (for example, there is no way of distinguishing between different orderings of the same set of levers). But as long as the levers still result in the calculated list of pyramid configurations, this error is unlikely to be noticed (at least not until after the student has graduated).

Input

The first line of input contains three integers n, k and t, where n ($1 \leq n \leq 40$) and k ($1 \leq k \leq 5$) are the number of levers and pyramids, respectively, and t ($1 \leq t \leq 3^k$) is the number of different pyramid configurations. Then follow t lines describing the distinct possible pyramid configurations. Each such line consists of a string of length k describing the configuration and an integer f ($1 \leq f \leq 2^n$) indicating the number of different lever settings which result in this configuration. The configuration is described using the letters '`A`', '`E`', and '`I`'. The jth character indicates the hieroglyph facing upwards on the jth pyramid: '`A`' for ankh, '`E`' for eye of Horus, and '`I`' for ibis.

The t configurations given are pairwise distinct, the sum of f-values over all t lines equals 2^n, and the input is such that at least one list of levers results in the given multiset of configurations.

Output

Output n lines describing a possible set of levers that gives rise to the given multiset of pyramid configurations. Each line describes one lever using a string of length k consisting of the symbols '`+`', '`-`', or '`0`'. In each such string, the jth symbol describes the action of this lever on the jth pyramid: '`+`' if the lever moves the pyramid clockwise, '`-`' if the lever moves it counterclockwise, and '`0`' if the lever does not move this pyramid.

If there is more than one possible solution, any one will be accepted.

Sample Input 1

```
2 3 4
EEE 1
EIA 1
IAE 1
AAA 1
```

Sample Output 1

```
+-0
+++
```

Sample Input 2

```
3 2 2
IA 4
AA 4
```

Sample Output 2

```
-0
00
00
```