Problem A. Yet Another Goat in the Garden

Time limit: 2 seconds
Memory limit: 256 megabytes

Once upon a time the world famous scientist Innokentiy decided to relax in the village. He has a garden of the triangular shape with sides a, b and c in his country site. The scientist often released his goat to walk there: unlike characters of some programming problems, the goat of Innokentiy was perfectly trained and didn't touch the grass and the vegetables in the garden.

Everything was changed when the new experiment of the scientist transmuted the goat into the black hole which had the shape of circle with a fixed radius r. Unlike the goat, the black hole was absorbing all the plants it touched while moving along the garden. Fortunately, Innokentiy has immediately figured out to set the innovative energy barriers along the perimeter of the garden. These barriers were designed so that no point of the black hole can cross them.

You are to calculate Innokentiy's losses. Find what part of the garden could be absorbed by the black hole in the worst case.

Input

The only line contains 4 integers a, b, c and r ($1 \le a, b, c, r \le 10^4$) — the lengths of the garden's sides and the radius of the black hole. It is guaranteed that there exists a nondegenerate triangle with sides a, b and c, and that the black hole can actually fit in the garden.

Output

Output a single real number — the biggest part of the garden that could be absorbed by the black hole. The answer will be considered correct if the absolute or relative error does not exceed 10^{-9}.

Examples

stdin	stdout
3 4 5 1	0.523598775598299
6 8 10 1	0.880899693899575

Problem B. Impossible to Guess

Time limit:	2 seconds
Memory limit:	256 megabytes

Andrew thought up a permutation $p_1, \ldots, p_n$ of n first positive integers and said it to Pavel so that he wouldn't forget it. This was not a very wise decision because Pavel suffers from memory disorders and can't remember the exact permutation, but for each its subsegment $[l, r]$ Pavel can quickly say a set of numbers $p_l, \ldots, p_r$ in ascending order.

Alex is very interested in the permutation thought up by Andrew so he wants to find it out from Pavel in secret from Andrew. Alex hasn't much time so he can ask only $\lceil \frac{n}{2} \rceil$ questions (where $\lceil x \rceil$ means the number x rounded up). Help him to find Andrew's permutation out.

Input and output

This is the interactive problem. Here your program must in the process of the solution exchange information with the jury's program. Please note that after outputting each message your program must flush the buffer, so that your output reaches the jury's program: for instance, calls of «`fflush(stdout)`» or «`cout.flush()`» do it in C++, «`System.out.flush()`» in Java, «`flush(output)`» in Pascal.

At the beginning the single integer n ($1 \le n \le 100$) — the length of the permutation — is sent your program. Then you can ask at most $\lceil \frac{n}{2} \rceil$ questions and after that give the final answer.

To ask a question you should output a letter «`Q`» first, and then two integers, separated by a space: l and r ($1 \le l \le r \le n$) — the indices of the first and the last elements of the subsegment you want to ask.

Jury's program will answer each of such questions, sending $(r - l + 1)$ integers separated by spaces: the elements of the permutation $p_l, \ldots, p_r$, sorted in the ascending order.

To give the final answer, you should output a letter «`A`» first, then n elements of the permutation separated by spaces — $p_1, \ldots, p_n$. After that you program must be terminated.

Examples

stdin	stdout
3	
	Q 1 1
3	
	Q 1 2
1 3	
	A 3 1 2

Note

Emply lines in the sample are given only for convenience, to make it clear in which order the messages are written. When solving the problem you must not output empty lines and jury's program won't output empty lines too.

Problem C. Lost Temple

Time limit: 2 seconds
Memory limit: 256 megabytes

Long time ago Andrew thought up a programming problem. Its main character was Indiana Jones who wanted to get into the temple of the lost ancient civilization but faced with a puzzle. There was a sculpture of a creature stretching out its hands near the temple's entrance, and on the temple's wall there was a suggestion to put into the left and into the right hands of the statue some non-zero amount of coins so that the product of those amounts is exactly k times greater than their sum. Of course if one does it wrong the doors to the temple won't open and nearby traps can kill or hurt Dr. Jones.

Unfortunately, this problem hasn't been given to the contest where Andrew wanted to publish it first. So you should solve it here and now. You should find all the ways to put coins into the sculpture's hands so that the doors to the temple will open.

Input

The only line contains the only integer k ($1 \le k \le 10^9$) — the number of times the product of the coins' amounts must be greater than their sum.

Output

In the first line output a single integer n — the number of different pairs of coins' amounts that open the doors to the temple.

In the each of the next n lines in any order output two integers separated by a space — the amounts of coins to put into the left and into the right hands of the statue, correspondingly. Each pair should be outputted exactly once; pairs that differ only by the order of its elements are considered different.

Examples

stdin	stdout
1	1 2 2
2	3 3 6 6 3 4 4

Problem D. Toy Soldiers

Time limit:	2 seconds
Memory limit:	256 megabytes

Petya loves toy soldiers very much. He has n soldiers, and i-th soldier is painted the a_i-th color.

Petya constantly doesn't like how his soldiers are colored so he takes one of them and repaints it. He does that m times, and Vova, watching on it, becomes interested when all the soldiers have the same color for the first time. Help Vova to answer his question.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of soldiers Petya has.

The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$) separated by spaces — the initial colors the soldiers were painted.

The third line contains a single integer m ($1 \le m \le 3 \cdot 10^5$) — the number of times Petya has repainted his soldiers.

The next m lines contains two integers each, separated by a space: k_j and x_j ($1 \le k_j \le n$, $1 \le x_j \le 10^9$) — the number of soldier and the number of color it has been repainted (possibly the same as before the repaintment).

Output

Output a single integer — the number of repaintments Petya has made before all his soldiers have the same color for the first time. In particular, if all soldiers had the same color before all Petya's actions, output «0». If such an event has never taken place at all, even after m-th repaintment, output «-1».

Examples

stdin	stdout
3 4 3 7 4 1 5 2 7 1 7 3 3	3
3 6 2 7 3 1 1 2 7 1 6	-1

Problem E. Just Change a Word

Time limit:	2 seconds
Memory limit:	256 megabytes

This problem tells about one of the methods of inventing new programming problems. One more method can be discovered reading the problem J.

There is one well-known problem, many times occured on very different sites and contests:

You are given a string of lowercase Latin letters. If two neighbour positions contain the same letters, this pair of letters can be removed. Can you get an empty string using such operations?

But you shouldn't solve this problem because, as we already said, it occured many times on different contests. The matter is that Pavel recently demonstrated Alex the easiest way of inventing new programming problems. One should just take a known problem and change one word in its statement. Now you have to solve the problem created just in this way. Here is the problem itself:

You are given a string of lowercase Latin letters. If two neighbour positions contain the *different* letters, this pair of letters can be removed. Can you get an empty string using such operations?

Input

The input contains the only non-empty string of lowercase Latin letters. Its length does not exceed $4 \cdot 10^5$.

Output

Output «`YES`», if you can get an empty string from the given one using only operation of removing two neighbour different letters. If it's impossible, output «`NO`».

Examples

stdin	stdout
aabccaba	YES
zzzzzzz	NO

Note

In the first sample, for example, such sequence of operations can be done: "`aabccaba`" → "`accaba`" → "`acba`" → "`ac`" → "".

Problem F. Two Envelopes

Time limit:	2 seconds
Memory limit:	256 megabytes

Mike has two envelopes. He thought up a random integer in the segment $[a, b]$ (he could think up every number from the segment with equal probability) and put exactly this amount of roubles into one of the envelopes and the double amount into the second one. Then he suggested Constantine to play the game: Constantine can open one envelope and then either take its contents or take the contents of another envelope, by his choice.

Constantine has opened one of these envelopes and has discovered c roubles there. Should he take another envelope, if he wants to maximize the expected prize?

Input

The only line contains three integers a, b and c ($1 \le a, b, c \le 10^9$), separated by spaces — the bounds of the segment from which Mike thought up his random number and the amount of roubles in the envelope opened by Constantine. It is guaranteed that the situation described by the input is possible.

Output

Output «`Take another envelope`», if it's advantageous for Constantine to take another envelope, and «`Stay with this envelope`», if it's advantageous for him to stay with the one he has already opened.

Examples

stdin	stdout
`3 5 3`	`Take another envelope`
`4 6 10`	`Stay with this envelope`

Problem G. Change-making Problem

Time limit:	2 seconds
Memory limit:	256 megabytes

There are $(n+1)$ types of coins in the country R, the cheapest of which has denomination 1 and each of the next types has denomination a_i times greater than the previous one. You need to pay the sum s using as few coins as possible. Of course you can use multiple coins of the same denomination.

Input

The first line contains two integers separated by a space: n and s $(1 \le n \le 10^5,\ 0 \le s \le 10^9)$ —- the number of coins' types, excluding the cheapest one, and the sum to pay.

The second line contains n integers separated by spaces: a_i $(2 \le a_i \le 10^9)$ — the number of times each of the next coins is more expensive than the previous one.

Output

Output a single integer — the minimum number of coins required to pay the sum s.

Examples

stdin	stdout
3 42 3 2 2	4
5 228 5 2 5 2 5	8

Problem H. Tony Hawk's Pro Skater

Time limit:	2 seconds
Memory limit:	256 megabytes

Alex loves the game Tony Hawk's Pro Skater very much. Some tasks in this game oblige the player to perform tricks by pressing certain combinations of buttons and to earn points for that: the more — the better.

Alex has learned how to perform n types of tricks. But to make the game more interesting, its authors provided the following: the reward for every trick decreases while player keeps performing it (but cannot become less than 1). Thus, if player performs the i-th trick for the first time, he earns a_i points, for the second time — $\max(a_i - b_i,\ 1)$ points and so on: k-th performance costs $\max(a_i - (k-1)\,b_i,\ 1)$ points. Performance of the trick of some type doesn't affect the costs of tricks of the other types.

The time in the game is limited so Alex is able to perform only m tricks. What is the maximal number of points he can gain?

Input

The first line contains two integers n and m ($1 \le n \le 10^5$, $0 \le m \le 10^9$) — the number of types of tricks Alex can perform and the number of tricks he has time to perform.

The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the initial costs of the tricks.

The next line contains n integers $b_1, \ldots, b_n$ ($1 \le b_i \le 10^9$) — the number of points by which the cost of the corresponding trick reduces at each its performance.

Output

Output a single integer — the maximal number of points which can be gained by performing m tricks.

Examples

stdin	stdout
3 6 9 7 17 1 2 3	67
3 7 9 7 17 2 1 4	68

Problem I. Map Coloring

Time limit: 2 seconds
Memory limit: 256 megabytes

Peter Godunov is ordered to make the map of Syberia. Syberia consists of n regions, some of which eccentrically border on each other. He wants to assign each of them some color but he has only k different paints. Moreover, according to the terms of reference, on the final version of the map two regions must have different color *if and only if* they are neighbour regions. It's not required to use all the paints. Help Peter to handle the task or he will be beheaded.

Input

The first line contains three integers separated by spaces: n, m and k ($1 \le n \le 1000$, $0 \le m \le \frac{n(n-1)}{2}$, $1 \le k \le n$) — the number of regions in Syberia, the number of pairs of neighbour regions and the number of paints Peter has.

The next m lines contain two integers each: x_j and y_j ($1 \le x_j, y_j \le n$, $x_j \ne y_j$) — the numbers of the neighbour regions. Each pair of regions occures at most once.

Output

Output n integers c_i separated by spaces ($1 \le c_i \le k$), where c_i is a color to paint the i-th region. If it will cost Peter his head, output the only number «-1».

Examples

stdin	stdout
3 2 2 1 2 2 3	1 2 1
4 4 2 1 2 2 3 3 4 4 1	1 2 1 2
4 4 3 1 2 1 3 1 4 2 3	-1

Problem J. Hyperdromes Strike Back

Time limit:	2 seconds
Memory limit:	256 megabytes

In the problem E one of the ways of inventing new programming problems is shown. Another one will be presented here.

On the NEERC 2012 contest there was, up to restatement, the following problem:

The string is called hyperdrome if its symbols can be shuffled so that this string becomes palindrome.

The number of hyperdromes in the string s of length n is the number of such pairs (l, r) $(1 \le l \le r \le n)$ that the substring $s_l \ldots s_r$ is a hyperdrome.

Given a string s of length at most 10^5, consisting of lowercase Latin letters, find how many hyperdromes it has.

We will not suggest you the problem from NEERC 2012, so we will just swap input and output.

You are given an integer k. Find a string of length at most 10^5, consisting of lowercase Latin letters, so that it has exactly k hyperdromes.

Input

The only line contains the only integer k $(1 \le k \le 2 \cdot 10^9)$ — the required number of hyperdromes in the string.

Output

Output a string of length at most 10^5, consisting of lowercase Latin letters, so that it has exactly k hyperdromes. It's guaranteed that the answer exists.

Examples

stdin	stdout
6	aaa
12	abadaba
5	azz

Problem K. Two Pirates

Time limit: 2 seconds
Memory limit: 256 megabytes

The famous pirate Jack Sparrow and not so famous pirate Ferrante Albrizzi, even less known as Malasorte, have robbed a Spanish trading schooner together and decided to divide the loot. The loot consists of n numbered items, the i-th of which costs a_i piastres.

The pirates decided to take items in rotation. Jack Sparrow takes first because he is more famous. But Ferrante Albrizzi is not interested in money since he has become a pirate not because of them, so he just always takes the first item that hasn't been taken yet. Jack Sparrow perfectly knows it and wants to act to maximize the cost of his part of the loot. Determine how much piastres their loot will cost.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of items in the loot.

The second line contains n integers separated by spaces: a_i ($1 \le a_i \le 10^9$) — the cost of the i-th item.

Output

Output in the only line two integers separated by a space: the cost of loot taken by Jack Sparrow and the cost of loot taken by Ferrante Albrizzi.

Examples

stdin	stdout
5 3 1 2 4 1	8 3
6 8 5 3 6 1 4	18 9

Problem L. Two Heads Are Better

Time limit: 2 seconds
Memory limit: 256 megabytes

World famous scientist Innokentiy has started studying computability theory and invented a new abstract executor that represents a tape of n cells, each of which contains a lowercase Latin letter. Two heads can move along the tape, each pointing at some cell. This device can execute the following instructions: to move the specified head to the left or to the right by one cell, to reverse the part of the tape between two heads, including the cells the heads point at, and to answer the request which symbol the specified head points at. Innokentiy asked you to help him to emulate the work of this device since the program he has written works too slow.

Input

The first line contains three integers separated by spaces: n, l and r ($1 \le n \le 10^5$, $1 \le l < r \le n$) — the number of cells in the tape and the initial positions of the left and the right head, correspondingly.

The second line contains n lowercase Latin letters, written in the cells.

The third line contains a single integer m ($1 \le m \le 3 \cdot 10^5$) — the number of queries.

In the next m lines there are the queries in the following form.

- `S X Y` — to move the head `X` in direction `Y`, where `X` can be `L` for the left head and `R` for the right one, and `Y` can be `L` for moving left or `R` for moving right.
- `R` — to reverse the part of the tape between the heads, including the cells the heads point at.
- `Q X` — to ask which character the head `X` points at, where `X` can be `L` for the left head and `R` for the right one.

It is guaranteed that the left head always remains to the left of the right head and that the heads don't move out of the tape.

Output

Output one string, containing all the answers for the queries of the last type. The k-th character of this string must be the answer to the k-th query of the type «`Q X`».

Examples

stdin	stdout
11 2 6 abracadabra 12 Q L Q R R Q L Q R S L R S R R Q L Q R R Q L Q R	baabcddc

Problem M. Construct a Permutation

Time limit: 2 seconds
Memory limit: 256 megabytes

Alex has recently learned the algorithm of finding the longest increasing subsequence of the array. The longest increasing subsequence of the array $a_1, \ldots, a_n$ is the sequence of numbers $a_{i_1}, \ldots, a_{i_m}$, where $i_1 < \ldots < i_m$ and $a_{i_1} < \ldots < a_{i_m}$, and m — the length of the longest increasing subsequence — is the greatest possible. For instance, in the array $\{5, 2, 3, 1, 4\}$ such a subsequence is $\{2, 3, 4\}$. Of course, the array can contain multiple longest increasing subsequences.

Alex has also figured out that the longest decreasing subsequence can be found by just reversing the array. To verify his guesses and the algorithm's work at all he decided to construct such a permutation that its longest increasing subsequence has length a and its longest decreasing one has length b. Moreover, the permutation must be the largest possible in order to test the performance of the algorithm more carefully. You are to find such a permutation.

Input

The only line contains two integers a and b ($1 \le a, b \le 500$) — the required length of the longest increasing and decreasing subsequences correspondingly.

Output

In the first line output a single integer n — the length of the permutation.

In the second line output the permutation of numbers from 1 to n, satisfying all the requirements.

The length of the permutation n must be the largest possible. If there are multiple solutions, output any of them.

Examples

stdin	stdout
2 1	2 1 2
2 2	4 2 4 1 3