Problem A. Alex Origami Squares

Input file:	`alex.in`
Output file:	`alex.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Alex is fond of origami — Japanese art of paper folding. Most origami designs start with a square sheet of paper. Alex is going to make a present for his mother. Present's design requires three equal square sheets of paper, but Alex has only one rectangular sheet. He is able to cut out squares of this sheet, but their sides should be parallel to the sides of the sheet. Help Alex to determine the maximum possible size of the paper squares he is able to cut out.

Input

The single line of the input file contains two integers h and w — the height and the width of the sheet of paper ($1 \le h, w \le 1000$).

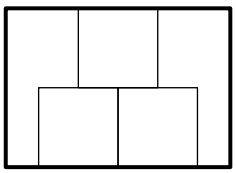
Output

Output a single real number — the maximum possible length of the square side. It should be possible to cut out three such squares of $h \times w$ sheet of paper, so that their sides are parallel to the sides of the sheet.

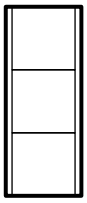
Your answer should be precise up to three digits after the decimal point.

Examples

alex.in	alex.out
210 297	105.0
250 100	83.333333



Example 1



Example 2

Problem B. Black and White

Input file:	`black.in`
Output file:	`black.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The jury has a great artistic idea — to create a rectangular panel out of a huge pile of black and white squares of the same size. The panel should have exactly b 4-connected areas made of black tiles, and w 4-connected areas made of white tiles.

Remember, a *4-connected area* of some color is a maximal set of the panel tiles such that:

- any two tiles of the area share the same color;
- for any two tiles of the area there is a tile sequence connecting them, such that any two consecutive tiles of the sequence share a common side.

In addition to the artistic idea, the jury has already developed a program that produces design of the panel. But since this problem is about art, any new solution is extremely important for the jury.

Input

The only line of the input file contains two integers b and w — number of black and white areas ($1 \le b, w \le 1000$).

Output

The first line of the output file should contain the picture sizes r and c — the number of rows and columns ($1 \le r, c \le 100\,000$). This line should be followed by r lines of c symbols each. Each symbol should be either '`@`' (for black tile) or '`.`' (for white one). There should be no more than 100 000 tiles in the panel.

Example

black.in	black.out
2 3	6 7 @@@@@@@ @.@@@@@ @@...@@ @@@@@@@ .... @@@@@@@

Problem C. Concatenation

Input file:	`concatenation.in`
Output file:	`concatenation.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Famous programmer Gennady likes to create new words. One way to do it is to concatenate existing words. That means writing one word after another. For example, if he has words “cat” and “dog”, he would get a word “catdog”, that could mean something like the name of a creature with two heads: one cat head and one dog head.

Gennady is a bit bored of this way of creating new words, so he has invented another method. He takes a non-empty prefix of the first word, a non-empty suffix of the second word, and concatenates them. For example, if he has words “tree” and “heap”, he can get such words as “treap”, “tap”, or “theap”. Who knows what they could mean?

Gennady chooses two words and wants to know how many different words he can create using his new method. Of course, being a famous programmer, he has already calculated the answer. Can you do the same?

Input

Two lines of the input file contain words chosen by Gennady. They have lengths between 1 and 100 000 characters and consist of lowercase English letters only.

Output

Output one integer — the number of different words Gennady can create out of words given in the input file.

Examples

concatenation.in	concatenation.out
cat dog	9
tree heap	14

Problem D. Distribution in Metagonia

Input file:	`distribution.in`
Output file:	`distribution.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

There are one hundred noble families in the country of Metagonia, and each year some of these families receive several ritual cubes from the Seer of the One. The One has several rules about cube distribution: if a family receives at least one cube, every prime divisor of the number of cubes received should be either 2 or 3, moreover if one family receives $a > 0$ cubes and another family in the same year receives $b > 0$ cubes then a should not be divisible by b and vice versa.

You are the Seer of the One. You know in advance how many cubes would be available for distribution for the next t years. You want to find any valid distribution of cubes for each of these years. Each year you have to distribute all cubes available for that year.

Input

The first line of input file contains a single integer t — the number of years to come ($1 \le t \le 1000$).

Each of the following t lines contains a single integer n_i — the number of cubes to distribute in i-th year ($1 \le n_i \le 10^{18}$).

Output

For each year i output two lines. The first line should contain m_i — the number of families that would receive at least one cube in i-th year ($1 \le m_i \le 100$). The second line should contain m_i integers — the number of cubes received by each family. The sum of these numbers should be equal to n_i.

Example

distribution.in	distribution.out
4 1 2 3 10	1 1 1 2 1 3 2 4 6

Problem E. Easy Arithmetic

Input file:	`easy.in`
Output file:	`easy.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Eva is a third-grade elementary school student. She has just learned how to perform addition and subtraction of arbitrary-precision integers. Her homework is to evaluate some expressions. It is boring, so she decided to add a little trick to the homework. Eva wants to add some plus and minus signs to the expression to make its value as large as possible.

Input

The single line of the input file contains the original arithmetic expression. It contains only digits, plus (‘`+`’) and minus (‘`-`’) signs.

The original expression is correct, that is:

- numbers have no leading zeroes;
- there are no two consecutive signs;
- the last character of the expression is a digit.

The length of the original expression does not exceed 1000 characters.

Output

Output a single line — the original expression with some plus and minus signs added. Output expression must satisfy the same correctness constraints as the original one. Its value must be as large as possible.

Examples

easy.in	easy.out
10+20-30	10+20-3+0
-3-4-1	-3-4-1
+10	+10

Problem F. Fygon

Input file:	`fygon.in`
Output file:	`fygon.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Frederick is a young programmer. He participates in all programming contests he can find and always uses his favorite programming language Fygon. Unfortunately, he often receives Time Limit Exceeded outcome, even when his algorithm is asymptotically optimal. That's because the Fygon interpreter is very slow. Nevertheless, Frederick likes Fygon so much, that he uses non-asymptotical optimizations to fit the solution into time limit. To make it easier, he asks you to write a program, which will be able to estimate the exact number of operations that his Fygon program makes.

For simplicity, we will assume that Fygon has only two statements. The first statement is `lag`. It substitutes almost any other statement. The second statement is a `for` loop:

```
for <variable> in range(<limit>):
    <body>
```

This means that `<variable>` iterates over values from 0 to `<limit>`-1. In Fygon `<variable>` is a lowercase letter from `a` to `z`, and `<limit>` is either already defined `<variable>` or a positive integer constant. The `<body>` of the loop is indented by four spaces and contains at least one statement.

The program receives the input in the variable `n`. This variable has special meaning and cannot be used as a loop variable.

Your task is to find the formula that calculates the number of performed `lag` operations by the given Fygon program, depending on the value of the variable `n`.

Input

The input file contains the Fygon program. No two loops use the same variable as iterators. Each variable used inside a `range` is either `n` or declared in some outer loop.

The program has at most 20 statements and at most 6 of them are loops. All integer constants are from 1 to 9.

Output

Output the formula for the number of performed `lag` operations depending on n. The length of the formula should be at most 100 characters (excluding spaces). The formula should correspond to the following grammar:

⟨Expression⟩ ::= ⟨Product⟩ ((‘`+`’ | ‘`-`’) ⟨Product⟩) *
⟨Product⟩ ::= ⟨Value⟩ (‘`*`’ ⟨Value⟩) *
⟨Value⟩ ::= ‘`n`’ | ⟨Number⟩ | ‘`-`’ ⟨Value⟩ | ‘`(`’ ⟨Expression⟩ ‘`)`’
⟨Number⟩ ::= [‘`0`’..‘`9`’] $^+$ (‘`/`’ [‘`0`’..‘`9`’] $^+$) $^?$

Example

fygon.in	fygon.out

`for i in range(n):`
`  for j in range(i):`
`  lag`
`for x in range(5):`
`  for y in range(n):`
`  for z in range(n):`
`  lag`
`  lag`
	`1/2 * n * (n-1) + 5 * (n*n + 1)`

Problem G. Graph

Input file:	`graph.in`
Output file:	`graph.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The sequence $a_1, a_2, \ldots, a_n$ is called a *permutation*, if it contains every integer from 1 to n.

The permutation of vertices $a_1, a_2, \ldots, a_n$ is a *topological sort* of a directed graph, if for every directed edge from u to v, vertex u comes before v in this permutation.

The permutation $a_1, a_2, \ldots, a_n$ is *lexicographically smaller* than the permutation $b_1, b_2, \ldots, b_n$, if there exists m such that $a_i = b_i$ for every $1 \le i < m$ and $a_m < b_m$.

Given a directed acyclic graph, add at most k directed edges to it in such a way, that the resulting graph still has no cycles and the lexicographically minimal topological sort of the graph is *maximum possible*.

Input

The first line of the input file contains three integers n, m and k — the number of vertices and directed edges in the original graph, and the number of directed edges, that you are allowed to add ($1 \le n \le 100\,000$; $0 \le m, k \le 100\,000$).

Each of the following m lines contains two integers u_i, v_i, describing directed edge from u_i to v_i ($1 \le u_i, v_i \le n$).

The graph has no cycles.

Output

The first line of the output file should contain n integers — the lexicographically minimal topological sort of the modified graph. The second line should contain a single integer x ($0 \le x \le k$) — the number of directed edges to add. The following x lines of the output should contain description of added directed edges in the same format as in the input file.

Examples

graph.in	graph.out
5 3 2 1 4 4 2 1 3	5 1 4 2 3 2 4 3 5 1
2 2 20 1 2 1 2	1 2 1 1 2

Problem H. Hash Code Hacker

Input file:	`hash.in`
Output file:	`hash.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

According to Java standard library documentation, the hash code of `String` is computed as

```
s[0]*31^(n-1) + s[1]*31^(n-2) + ... + s[n-1]
```

Here `s[i]` is the i-th character of the string, `n` is the length of the string, and `^` indicates exponentiation. Computation uses signed 32-bit integers in two's complement form.

Heather is going to hack the servers of Not Entirely Evil Recording Company (NEERC). To perform an attack she needs k distinct query strings that have equal hash codes. Unfortunately, NEERC servers accept query string containing lower- and uppercase English letters only.

Heather hired you to write a program that generates such query strings for her.

Input

The single line of the input file contains integer k — the number of required query strings to generate ($2 \le k \le 1000$).

Output

Output k lines. Each line should contain a single query string. Each query string should be non-empty and its length should not exceed 1000 characters. Query string should contain only lower- and uppercase English letters. All query strings should be distinct and should have equal hash codes.

Example

hash.in	hash.out
4	edHs mENAGeS fEHs edIT

Problem I. Insider's Information

Input file:	`insider.in`
Output file:	`insider.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Ian works for a rating agency that publishes ratings of the best universities. Irene is a journalist who plans to write a scandalous article about the upcoming rating.

Using various social engineering techniques (let's not get into more details), Irene received some insider's information from Ian.

Specifically, Irene received several triples (a_i, b_i, c_i), meaning that in the upcoming rating, university b_i stands between universities a_i and c_i. That is, either a_i comes before b_i which comes before c_i, or the opposite. All triples told by Ian are consistent — let's say that actual rating *satisfies* them all.

To start working on the first draft of the future article, Irene needs to see at least some approximation to the actual rating. She asked you to find a proposal of a rating in which at least half of the triples known by Irene are satisfied.

Input

The first line contains integers n and m, the number of rated universities, and the number of triples given to Irene by Ian ($3 \le n \le 100\,000$; $1 \le m \le 100\,000$).

Each of the next m lines contains three distinct integers a_i, b_i, c_i — the universities making a triple ($1 \le a_i, b_i, c_i \le n$).

Output

Output the proposal of a rating from the first university to the last one. The proposal rating should satisfy at least $\frac{m}{2}$ triples. If there are many such proposals, output any one of them.

Example

insider.in	insider.out
4 3 1 2 3 1 2 3 1 4 3	4 3 2 1

In the example above, the first two triples are satisfied whereas the last one is not. Therefore, at least half of all triples are satisfied.

Problem J. Journey to the "The World's Start"

Input file:	journey.in
Output file:	journey.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Jerry Prince is the fourth grade student and he goes to New-Lodnon to visit the most popular amusement park "The World's Start".

An airport he arrives at is next to the first stop of the metro line. This line has n stops and "The World's Start" is on the last of them. The metro of New-Lodnon is pretty fast so you may assume that you can get from a stop to the next one in just one minute.

Jerry needs a travel card to use the metro. Each travel card has a range r and a price p. With a travel card of range r Jerry may travel no more than r stops at once. Therefore, if Jerry enters metro at the stop i he should exit on one of the stops from $i - r$ to $i + r$ inclusive. It takes d_i minutes to exit and reenter metro at i-th stop. There is no time required to enter the first stop or exit the last one.

Jerry is not very rich but he has some spare time, so he decided to buy the cheapest travel card that will allow him to travel from the first metro stop to the last one in no more than t minutes.

Input

The first line of the input file contains two integers n and t — the number of stops and the maximum possible time ($2 \le n \le 50\,000$; $n - 1 \le t \le 10^9$).

The second line contains $n - 1$ integers p_r — the prices of travel cards with range $r = 1 \dots n - 1$ ($1 \le p_r \le 100\,000$)

The third line contains $n - 2$ integers d_i — the number of minutes required to reenter metro at stop $i = 2 \dots n - 1$ ($1 \le d_i \le 10^5$).

Output

Output a single integer p — the lowest possible price of one travel card that allows Jerry to travel from the first to the last stop in no more than t minutes.

Example

journey.in	journey.out
4 4 1 2 3 1 4	2

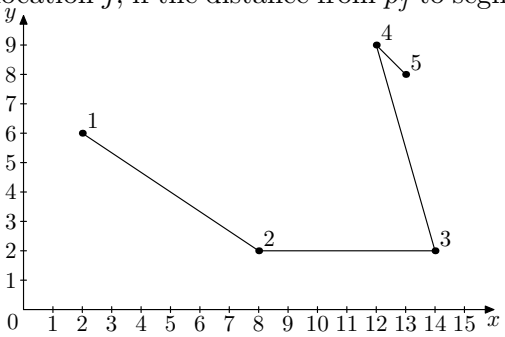
Problem K. Kingdom Trip

Input file:	`kingdom.in`
Output file:	`kingdom.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

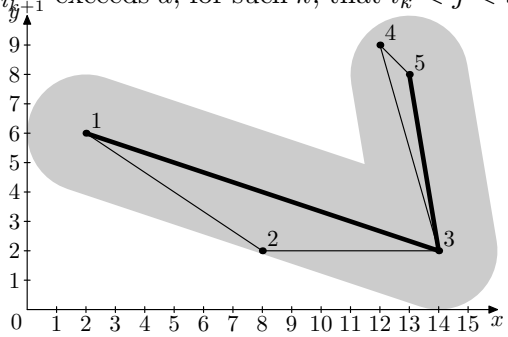
Once upon a time, there was a kingdom ruled by a wise king. After forty three years of his reign, by means of successful military actions and skillful diplomacy, the kingdom became an infinite flat two-dimensional surface. This form of the kingdom greatly simplified travelling, as there were no borders.

A big holiday was planned in the kingdom. There were n locations for people to gather. As the king wanted to have a closer look at his people, he ordered to make a trip through these locations. He wanted to give a speech in each of these locations. Initially his trip was designed as a polygonal chain p: $p_1 \to p_2 \to \ldots \to p_n$.

Not only the king was wise, but he was old, too. Therefore, his assistants came up with an idea to skip some locations, to make the king to give as few speeches as possible. The new plan of the trip has to be a polygonal chain consisting of some subsequence of p: starting at p_1 and ending at p_n, formally, $p_{i_1} \to p_{i_2} \to \cdots \to p_{i_m}$, where $1 = i_1 < i_2 < \cdots < i_m = n$. Assistants know that the king wouldn't allow to skip location j, if the distance from p_j to segment $p_{i_k} \to p_{i_{k+1}}$ exceeds d, for such k, that $i_k < j < i_{k+1}$.



Original route



New route

Help the assistants to find the new route that contains the minimum possible number of locations.

Input

The first line of the input file contains two integers n and d — the number of locations in the initial plan of the trip and the maximum allowed distance to skipped locations ($2 \le n \le 2000$; $1 \le d \le 10^6$).

The following n lines describe the trip. The i-th of these lines contains two integers x_i and y_i — coordinates of point p_i. The absolute value of coordinates does not exceed 10^6. No two points coincide.

Output

Output the minimum number of locations the king will visit. It is guaranteed that the answer is the same for $d \pm 10^{-4}$.

Example

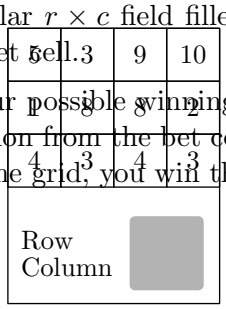
kingdom.in	kingdom.out
5 2 2 6 8 2 14 2 12 9 13 8	3

Problem L. Lucky Chances

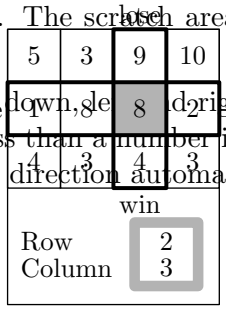
Input file:	`lucky.in`
Output file:	`lucky.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Lucky Chances is a lottery game. Each lottery ticket has a play field and a scratch area. The play field is a rectangular $r \times c$ field filled with numbers. The scratch area hides row and column numbers that specify the bet cell.

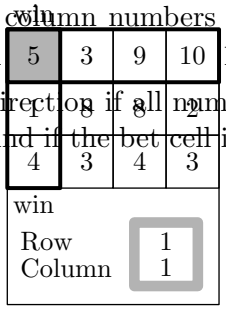
There are four possible winning directions: up, down, left and right. You win a direction if all numbers in this direction from the bet cell are strictly less than a number in the bet cell. And if the bet cell is on the edge of the grid, you win the corresponding direction automatically!



Unscratched ticket



Scratched ticket 1



Scratched ticket 2

Larry wants to choose the ticket that has maximum total number of winning directions for all possible bet cells. Write a program that determines this number for the given grid.

Input

The first line of the input file contains two integers r and c — the number of rows and columns in the grid ($1 \le r, c \le 100$).

The following r lines contain c integers each — the numbers printed on the grid. Each number is positive and does not exceed 1000.

Output

Output a single integer w — the total number of winning directions for the given grid.

Example

lucky.in	lucky.out
3 4 5 3 9 10 1 8 8 2 4 3 4 3	25