Problem A. Bike Roads

Input file:	`bike.in`
Output file:	`bike.out`
Time limit:	1 second
Memory limit:	64 megabytes

Andrew lives at countryside. The area he lives at has two bike roads, each of which has the form of a circle with radius r. Roads have no common points.

Andrew's house is located at one of the roads, and his school is located at the other one. Each day Andrew rides his bike from home to school and back. He has noticed that riding along the road is easier than riding by the ground. When riding along the road Andrew's speed is u, and when riding by the ground his speed is v. Now Andrew wonders what minimal time he needs to get from his house to school.

Let us introduce the coordinate system so that the center of the bike road where Andrew's house is located were $(0, 0)$, and the center of the bike road where his school is located were $(0, d)$. The radius of each road is r. Andrew's house is located at a point (x_1, y_1), and his school is located at a point (x_2, y_2). His speed by the road is u, and his speed by the ground is v.

Input

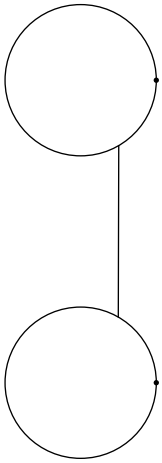
Input file contains eight floating point numbers: d, r, x_1, y_1, x_2, y_2, u and v ($1 \le r \le 100$, $2r < d \le 100$, $1 \le v < u \le 10$, $\sqrt{x_1^2 + y_1^2} = r$, $\sqrt{x_2^2 + (y_2 - d)^2} = r$). All equalities are up to 10^{-9}.

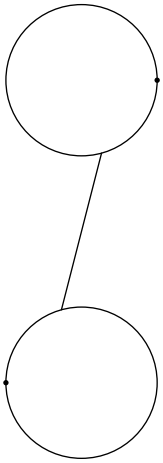
Output

Output one floating point number — the minimal time Andrew needs to get from his house to school. Your answer must be accurate up to 10^{-6}.

Example

bike.in	bike.out
20 5 5 0 5 20 2 1	16.5757337181
20 5 -5 0 5 20 2 1	17.2040517249





Problem B. Diversion

Input file:	`diversion.in`
Output file:	`diversion.out`
Time limit:	1 second
Memory limit:	64 megabytes

The kingdom of Farland has n cities connected by m bidirectional roads. Some of the roads are paved with stone, and others are just country roads. The capital of the kingdom is the city number 1. The roads are designed in such a way that it is possible to get from any city to any other using only roads paved with stone, and the number of stone roads is minimal possible. The country roads were designed in such a way that if any stone road is blocked or destroyed it is still possible to get from any city to any other by roads.

Let us denote the number of stone roads needed to get from city u to city v as $s(u, v)$. The roads were created long ago and follow the strange rule: if two cities u and v are connected by a road (no matter, stone or country), then either $s(1, u) + s(u, v) = s(1, v)$ or $s(1, v) + s(v, u) = s(1, u)$.

The king of Edgeland is planning to attack Farland. He is planning to start his operation by destroying some roads. Calculations show that the resources he has are enough to destroy one stone road and one country road. The king would like to destroy such roads that after it there were at least two cities in Farland not connected by roads any more.

Now he asks his minister of defense to count the number of ways he can organize the diversion. But the minister can only attack or defend, he cannot count. Help him!

Input

The first line of the input file contains n and m — the number of cities and roads respectively ($3 \le n \le 20\,000$, $m \le 100\,000$). The following m lines describe roads, each line contains three integer numbers — the numbers of cities connected by the corresponding road, and 1 for a stone road or 0 for a country road. No two cities are connected by more than one road, no road connects a city to itself.

Output

Output one integer number — the number of ways to organize the diversion.

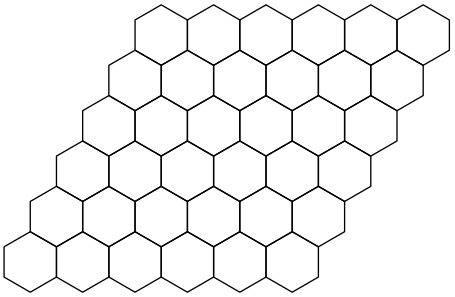
Example

diversion.in	diversion.out
6 7 1 2 1 2 3 1 1 4 0 3 4 1 4 5 1 3 6 0 5 6 1	4

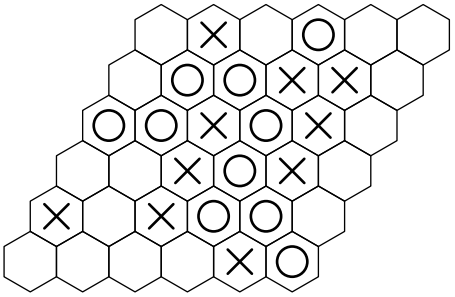
Problem C. Hex

Input file:	`hex.in`
Output file:	`hex.out`
Time limit:	1 second
Memory limit:	64 megabytes

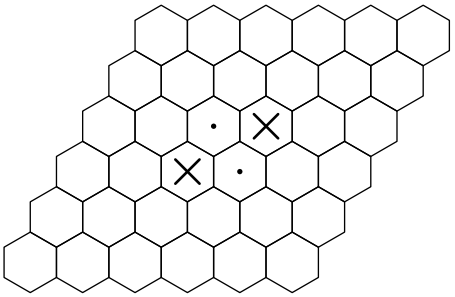
The game of hex is played on a rhombic field of size $n \times n$ divided to hexagonal cells. An example of the field for $n = 6$ is shown on the picture below.



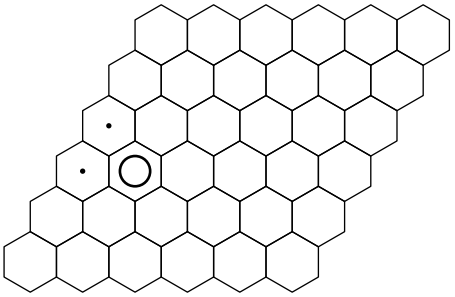
Two players, one playing for noughts and one for crosses, make moves in turn. A move consists of putting a player's piece to one of the free cells. Two cells are considered adjacent if they have a common side. The goal of noughts is to make a chain of adjacent cells from the left edge of the field to the right edge of the field, the goal of crosses is to make such chain from the top edge to the bottom edge. A picture below shows the game won by noughts.



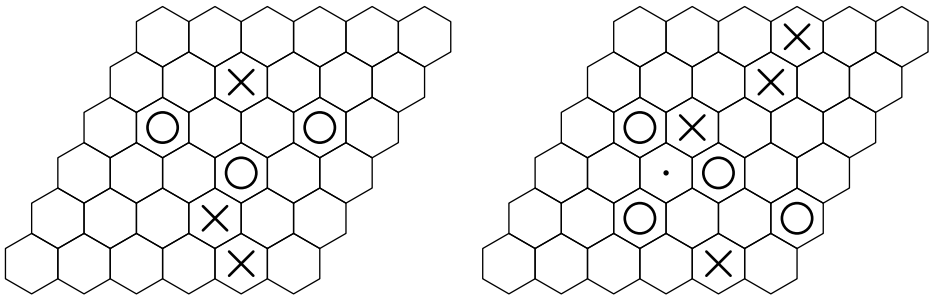
The gameplay of hex strongly depends on so called *templates*. In this problem we will consider two simple templates. The *two-bridge* template is the situation when two player's pieces can be connected in two ways in one move. Such pieces can be considered connected. A picture below shows the two-bridge for crosses. Note that the cells marked by dots must not be occupied. If noughts play to one of these cells, crosses reply to another cell connecting their pieces. The two cells marked by dots are called *bridge cells*.



Another simple template is a similar construction at the edge of the board, called *template II*. A piece in the second row at the player's edge, with no pieces in the two cells at the edge adjacent to it, can always be connected to the edge. A picture below shows template II for noughts.



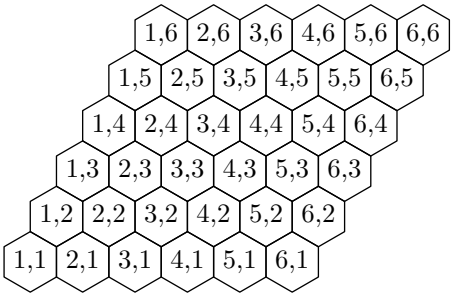
The game is said to be won up to simple templates by one of the players if there is a sequence $p_1, p_2, \ldots, p_k$ of his pieces such that p_1 and p_k are at his different edges or connected to them by template II, for all i pieces p_i and p_{i+1} are either adjacent or connected to each other by a two-bridge, and no two-bridges share a bridge cell. A picture on the left below shows a game won by noughts up to simple templates. A game on the right is not won by noughts up to simple templates yet, because a cell marked by a dot is shared by two two-bridges.



Given the position of the field, find out whether the game is won up to simple templates by one of the players.

Input

The first line of the input file contains two integer numbers: n and m — the size of the field and the number of pieces on the field ($3 \le n \le 10\,000$, $1 \le m \le 30\,000$). The following m lines describe pieces. Each piece is described by three integer numbers: x, y and s — the coordinates of the piece and the player who owns the piece. Coordinates range from 1 to n, s is 0 for noughts and 1 for crosses. The position need not be a valid game position, so the number of pieces of different players can be different. No two pieces occupy the same cell. The coordinate system is shown on the picture below, each cell is marked as x, y.



Output

Output "`noughts`" if the game is won by noughts up to simple templates, "`crosses`" if the game is won by crosses up to simple templates and "`none`" if the game is not won up to simple templates yet.

Example

hex.in	hex.out
6 6 2 4 0 4 3 0 5 4 0 3 5 1 4 2 1 5 1 1	noughts
6 8 2 4 0 3 2 0 4 3 0 6 2 0 3 4 1 4 5 1 4 6 1 5 1 1	none

Problem D. Monotone Gray Code

Input file:	`monotone.in`
Output file:	`monotone.out`
Time limit:	1 second
Memory limit:	64 megabytes

A Gray code of order n is the sequence of all binary vectors of length n such that two adjacent vectors differ in exactly one bit. An example of the Gray code for $n = 3$ is shown on the following figure.

000
001
011
010
110
111
101
100

The Gray code is called *monotone* if for any k no vector with $k + 2$ or more ones ever appears before a vector with k ones. For example, the code above is not monotone, because the vector `111` appears before the vector `100`. An example of the monotone Gray code for $n = 3$ is shown below.

000
001
011
010
110
100
101
111

It can be shown that for any n there exists a monotone Gray code. Given n find the monotone Gray code for binary vectors of length n.

Input

The input file contains one integer number n ($1 \le n \le 16$).

Output

Output the monotone Gray code for binary vectors of length n, one vector on a line.

Example

monotone.in	monotone.out
3	000 001 011 010 110 100 101 111

Problem E. Nice Sequence

Input file:	`nice.in`
Output file:	`nice.out`
Time limit:	1 second
Memory limit:	64 megabytes

Let us consider the sequence $a_1, a_2, \ldots, a_n$ of non-negative integer numbers. Denote as $c_{i,j}$ the number of occurrences of the number i among $a_1, a_2, \ldots, a_j$. We call the sequence *k-nice* if for all $i_1 < i_2$ and for all j the following condition is satisfied: $c_{i_1,j} \geq c_{i_2,j} - k$.

Given the sequence $a_1, a_2, \ldots, a_n$ and the number k, find its longest prefix that is k-nice.

Input

The first line of the input file contains n and k ($1 \leq n \leq 200\,000$, $0 \leq k \leq 200\,000$). The second line contains n integer numbers ranging from 0 to n.

Output

Output the greatest l such that the sequence $a_1, a_2, \ldots, a_l$ is k-nice.

Example

nice.in	nice.out
10 1 0 1 1 0 2 2 1 2 2 3	8
2 0 1 0	0

Problem F. Integer Packing

Input file:	`packing.in`
Output file:	`packing.out`
Time limit:	1 second
Memory limit:	64 megabytes

When transmitting data by network it is important to pack it efficiently since this allows to optimize bandwidth usage. We will describe a decoding procedure in one efficient way of packing integer numbers. It uses variable length and has prefix property to ensure correct decoding. Your task will be to implement the optimal encoding method.

The method is applicable for transmitting integer numbers from -2^{63} to $2^{63}-1$ (`long` type in Java). Packed integer occupies from 1 to 9 bytes, depending on its value.

Decoding is performed in the following way. First the leading byte of the number is received. Looking at its bits from higher to lower, find the first zero. Let q be the number of these leading ones (if the first byte is `0xff`, $q=8$). q means the number of bytes to follow. If $q=8$, these eight bytes are a standard big-endian (higher byte first) representation of an integer number.

In the other case the following operation is performed: if the $q+2$-nd bit of the first byte is zero (bits are numbered from 1, higher bits first, if $q=7$, the highest bit of the second byte is considered), leading ones of the first byte are replaced by zeroes, and the number is padded on the left by zero bytes to eight bytes. In the other case, the $q+1$-st bit (it is always zero) is replaced with one, and the number is padded on the left with `0xff` bytes to eight bytes. The result is a standard big-endian representation of an integer number.

Given several integer number, you must encode each of them so that the length of the encoded number were minimal possible and it decoded to the original number.

Input

The first line of the input file contains n — the number of test cases ($1 \le n \le 10\,000$). The following n lines contain one number between -2^{63} and $2^{63}-1$, inclusive, each.

Output

Output n lines, each line must contain an encoded version of the corresponding number written as the sequence of hexadecimal digits, higher bytes first.

Example

packing.in	packing.out
9	00
0	01
1	02
2	03
3	8064
100	81f4
500	cf4240
1000000	7f
-1	bf9c
-100	

Problem G. Rectangular Polygon

Input file:	`polygon.in`
Output file:	`polygon.out`
Time limit:	1 second
Memory limit:	64 megabytes

A rectangular polygon is a polygon whose edges are all parallel to the coordinate axes. The polygon must have a single, non-intersecting boundary. No two adjacent sides must be parallel.

Johnny has several sticks of various lengths. He would like to construct a rectangular polygon. He is planning to use sticks as horizontal edges of the polygon, and draw vertical edges with a pen.

Now Johnny wonders, how many sticks he can use. Help him, find the maximal number of sticks that Johnny can use. He will use sticks only as horizontal edges.

Input

The first line of the input file contains n — the number of sticks ($1 \le n \le 100$). The second line contains n integer numbers — the lengths of the sticks he has. The length of each stick doesn't exceed 200.

Output

Print l — the number of sticks Johnny can use at the first line of the output file. The following $2l$ lines must contain the vertices of the rectangular polygon Johnny can construct. Vertices must be listed in order of traversal. The first two vertices must be the ends of a horizontal edge. If there are several solution, output any one. Vertex coordinates must not exceed 10^9.

If no polygon can be constructed, output $l = 0$.

Example

polygon.in	polygon.out
4 1 2 3 5	3 0 0 1 0 1 1 3 1 3 2 0 2
4 1 2 4 8	0
4 1 1 1 1	4 0 0 1 0 1 1 2 1 2 -2 1 -2 1 -1 0 -1

In the first example Johnny uses a stick of length 1 for $(0, 0)-(1, 0)$ edge, a stick of length 2 for $(1, 1)-(3, 1)$ edge and a stick of length 3 for $(3, 2) - (0, 2)$ edge. There is no way to use all four sticks.

Problem H. SETI

Input file:	`seti.in`
Output file:	`seti.out`
Time limit:	1 second
Memory limit:	64 megabytes

Amateur astronomers Tom and Bob try to find radio broadcasts of extraterrestrial civilizations in the air. Recently they received some strange signal and represented it as a word consisting of small letters of the English alphabet. Now they wish to decode the signal. But they do not know what to start with.

They think that the extraterrestrial message consists of words, but they cannot identify them. Tom and Bob call a subword of the message a *potential word* if it has at least two non-overlapping occurrences in the message.

For example, if the message is "`abacabacaba`", "`abac`" is a potential word, but "`acaba`" is not because two of its occurrences overlap.

Given a message m help Tom and Bob to find the number of potential words in it.

Input

Input file contains one string that consists of small letters of the English alphabet. The length of the message doesn't exceed 10 000.

Output

Output one integer number — the number of potential words in a message.

Example

seti.in	seti.out
abacabacaba	15

Problem I. Sum vs Product

Input file:	`sump.in`
Output file:	`sump.out`
Time limit:	1 second
Memory limit:	64 megabytes

Peter has just learned mathematics. He learned how to add, and how to multiply. The fact that $2 + 2 = 2 \times 2$ has amazed him greatly. Now he wants find more such examples.

Peters calls a collection of numbers *beautiful* if the product of the numbers in it is equal to their sum. For example, the collections $\{2, 2\}$, $\{5\}$, $\{1, 2, 3\}$ are beautiful, but $\{2, 3\}$ is not.

Given n, Peter wants to find the number of beautiful collections with n numbers. Help him!

Input

The first line of the input file contains n ($2 \le n \le 500$).

Output

Output one number — the number of the beautiful collections with n numbers.

Example

sump.in	sump.out
2	1
5	3

The collections in the last example are: $\{1, 1, 1, 2, 5\}$, $\{1, 1, 1, 3, 3\}$ and $\{1, 1, 2, 2, 2\}$.

Problem J. Superposition

Input file:	`superposition.in`
Output file:	`superposition.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Tony is writing financial software. One of the components he is writing now will have to deals with risk and income graphs of various market instruments. Now he needs to plot the graph of a superposition of two continuous piecewise linear functions.

Tony is given two functions: $f(x)$ and $g(x)$ and needs to find the explicit representation of the function $g(f(x))$.

Input

The input file consists of two blocks, one describes f and another describes g.

Each blocks starts with l — the number of intervals of linearity of a function ($1 \le l \le 1000$, $l \ne 2$).

If $l > 2$, the following $l - 1$ lines contain the breaking points of the function. Each line contains two integer numbers: x_i, y_i (i from 1 to $l-1$, $x_i < x_{i+1}$). The function $a(x)$ described by the block is defined as follows:

$$a(x) = \begin{cases} y_1, \text{ if } x < x_1; \\ y_1 + k_1(x - x_1), \text{ if } x_1 \le x < x_2; \\ \dots \\ y_{l-2} + k_{l-2}(x - x_{l-2}), \text{ if } x_{l-2} \le x < x_{l-1}; \\ y_{l-1}, \text{ if } x \ge x_{l-1}. \end{cases}$$

Here $k_i = (y_{i+1} - y_i)/(x_{i+1} - x_i)$.

If $l = 1$ the following line contains one integer number y, the function described is $a(x) = y$.

All coordinates in the input file do not exceed 10^9 by their absolute values.

Output

Output the function $g(f(x))$ in the same format as the functions in the input file. The number of intervals of linearity in the description must be minimal possible. It is guaranteed that the number of intervals of linearity of the resulting function will not exceed 100 000. Output floating point numbers with absolute or relative precision of at least 10^{-8}.

Example

superposition.in	superposition.out
3 0 0 2 3 4 0 0 1 3 2 0	4 0 0 0.666666666667 3 1.333333333333 0