Problem A. Art

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Famous abstract artist Peter painted a peculiar piece of art involving a grid of cells. He's not quite happy with it. Some of the colors in his painting are touching, and he doesn't like that. He would like to use a dab of whiteout to separate any different colors that are touching. The painting consists of an $r \times c$ grid of cells where each cell is painted some color. Two colored cells are touching if they share a border edge. Cells sharing only a corner are not considered touching.

Cells are labelled with lowercase letters to denote colors. Peter can replace some of the cells with '#' representing whiteout. For example:

```
aaab
aabb
aaab
cccc
```

is transformed to

```
aa#b
a#bb
aa#b
##c#
```

Peter has separated all of the colors using only 6 dabs of whiteout. In order to save whiteout, Peter would like to minimize the number of cells he must dab.

But Peter, in the process of trying to save his painting, may accidentally paint over all of his favorite color! So while using the minimum number of dabs of whiteout, he would like to know which colors could possibly be retained in the painting.

For example the painting:

```
bab
bbb
```

Can be fixed using one dab:

```
b#b
bbb
```

It is possible to save the color `b` with a single dab, but it is not possible to use only a single dab and retain the color `a`.

Determine this minimum number of dabs and which colors are possible to save.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with two space-separated integers r and c ($1 \le r, c \le 100$), where r is the number of rows in the painting, and c is the number of columns.

Each of the next r lines will contain a string of length exactly c, consisting of only lower case letters, or the symbol '#' representing a cell that has already been whited out.

Output

Output exactly two lines. On the first line output a single integer, which is the minimum number of additional cells that must be whited out, not counting those that are already whited out. On the second line output a string of lower case letters in increasing alphabetic order. These should be the colors in the painting that can be retained by whiting out a minimum number of cells.

Example

standard input	standard output
2 2 ab ba	2 ab
2 3 aba aaa	1 a
3 4 #a## bcde ##f#	2 abef

Note

In the first example, `a` can be retained with 2 dabs:

```
a#
#a
```

And `b` can be retained with 2 dabs:

```
#b
b#
```

So, both `a` and `b` can be retained with 2 dabs, just not at the same time.

Problem B. Balanced or Bad?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Rock-paper-scissors is a zero-sum hand game usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are "rock", "paper", and "scissors". The game has only three possible outcomes other than a tie: a player who decides to play rock will beat another player who has chosen scissors ("rock crushes scissors") but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors ("scissors cut paper"). If both players choose the same shape, the game is tied and is usually immediately replayed to break the tie.

Recently, there is a upgraded edition of this game: rock-paper-scissors-Spock-lizard, in which there are totally five shapes. The rule is simple: scissors cuts paper; paper covers rock; rock crushes lizard; lizard poisons Spock; Spock smashes scissors; scissors decapitates lizard; lizard eats paper; paper disproves Spock; Spock vaporizes rock; and as it always has, rock crushes scissors.

Both rock-paper-scissors and rock-paper-scissors-Spock-lizard are balanced games. Because there does not exist a strategy which is better than another. In other words, if one chooses shapes randomly, the possibility he or she wins is exactly 50 percents no matter how the other one plays (if there is a tie, repeat this game until someone wins). Given an integer N, representing the count of shapes in a game. You need to find out if there exist a rule to make this game balanced.

Input

The first line of input contains an integer t, the number of test cases ($1 \le T \le 999$). t test cases follow. For each test case, there is only one line with an integer N ($2 \le N \le 1000$), as described above.

Output

For each test cases, output "`Balanced`" if there exist a rule to make the game balanced, otherwise output "`Bad`".

Example

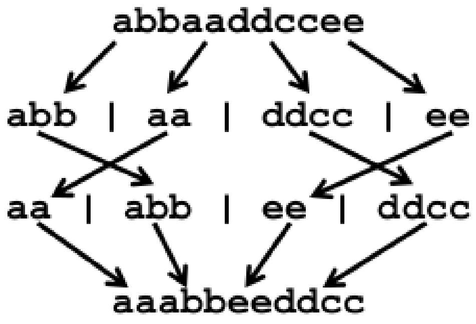
standard input	standard output
3 2 3 5	Bad Balanced Balanced

Problem C. Cut The Paper

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Lisa has been given a paper strip with a string of little letters written on it. She would like to rearrange the letters. She does this by making some number of cuts between letters and then rearranging the strips of paper.

Lisa likes order, so she would like the resulting strip of paper to be bitonic. That is, there should be some character position in the resulting string where the characters up to and including that position are alphabetically non-decreasing and all characters after and including that position are alphabetically non-increasing. Consider this example:



The resulting string in the above example is bitonic. Consider the first ‘`e`’. The string “`aaabbe`” is non-decreasing, and the string “`eeddcc`” is non-increasing. Lisa achieved this with three cuts. Note that any string which is monotonic (uniformly nondecreasing or nonincreasing) is also bitonic.

Determine the minimum number of cuts that Lisa needs in order to make his string bitonic.

Input

Input will consist of a single line containing a string s ($1 \le |s| \le 50$) which consists only of lower-case letters. This is the original string on the strip of paper given to Tito.

Output

Output a single integer, which is the minimum number of required cuts.

Example

standard input	standard output
abbaaddccee	3

Problem D. Development of Roads

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Roads in a city play important roles in development of the city. Once a road is built, people start their living around the road. A broader road has a bigger capacity, that is, people can live on a wider area around the road.

Interstellar Conglomerate of Plantation and Colonization (ICPC) is now planning to develop roads on a new territory. The new territory is a square and is completely clear. ICPC has already made several plans, but there are some difficulties in judging which plan is the best.

Therefore, ICPC has collected several great programmers including you, and asked them to write programs that compute several metrics for each plan. Fortunately your task is a rather simple one. You are asked to compute the area where people can live from the given information about the locations and the capacities of the roads.

The following figure shows the first plan given as the sample input.

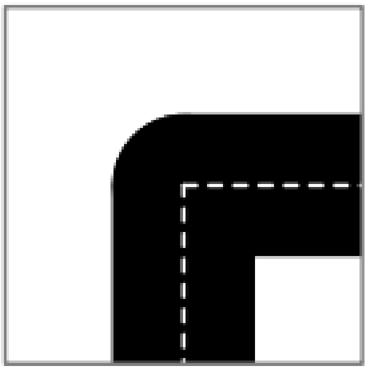


Figure 1 shows the first plan given as the sample input.

Input

The input consists of a number of plans. The first line of each plan denotes the number n of roads ($n \leq 50$), and the following n lines provide five integers x_1, y_1, x_2, y_2, and r, separated by a space. (x_1, y_1) and (x_2, y_2) denote the two endpoints of a road ($-15 \leq x_1, y_1, x_2, y_2 \leq 15$), and the value r denotes the capacity that is represented by the maximum distance from the road where people can live ($r \leq 10$).

The end of the input is indicated by a line that contains only a single zero.

The territory is located at $-5 \leq x, y \leq 5$.

You may assume that each road forms a straight line segment and that any lines do not degenerate.

Output

Print the area where people can live in one line for each plan. You may print an arbitrary number of digits after the decimal points, provided that difference from the exact answer is not greater than 0.01.

Example

standard input	standard output

2
0 -12 0 0 2
0 0 12 0 2
0
	39.14159

Problem E. Erasing an Edge

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given a weighted simple connected undirected graph G of N vertices and M edges. The vertices are numbered by sequential integers between 1 and N, and the edges are numbered by sequential integers between 1 and M. The i-th edge connects the vertces u_i and v_i ($u_i < v_i$) and its weight is w_i.

Consider the weight $W(X)$ of subgraph X as sum of weights of all edges in the subgraph.

Your task is to remove one edge from the graph such as graph will be split on two connected components A and B (B is empty when graph is still connected after the edge removal) and $|W(A) - W(B)|$ is minimized.

If multiple solution exists, choose one with smallest u, if ambiguity is not resolved, choose an edge with smallest v.

Input

First line of the input contains two integers N and M ($2 \leq N \leq 10^5$, $1 \leq M \leq 10^5$). Then M lines follow. Each of M lines contains three integers u, v and w ($1 \leq u < v \leq N$, $1 \leq w \leq 10^9$) — ids of connected vertices and weight, respectively. It is guaranteed that no pair (u, v) repeatc in the input and that graph is connected.

Output

Print two integers — a required edge.

Example

standard input	standard output

`5 4`
`1 2 1`
`2 3 10`
`3 4 5`
`4 5 1`
	`2 3`

`10 11`
`1 2 1`
`2 3 10`
`1 5 2`
`3 4 7`
`3 5 9`
`5 6 8`
`6 7 5`
`6 8 3`
`7 8 12`
`7 9 1`
`9 10 8`
	`5 6`

Problem F. Floors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given a building of N floors, numbered from 1 to N. A tourist starts on first floor and at each step can move one floor up or one floor down (if it is possible, i.e. tourist shall move up from floor 1 and shall move down from floor N). Calculate number of different M-step routes for tourist to visit each floor of the building atleast once.

Input

Input contains two integers N and M ($2 \le N \le 10^5$, $1 \le M \le 10^5$) — number of floors and number of steps, respectively.

Output

Print one integer — number of M-step routes visiting all floors modulo $10^9 + 7$.

Examples

standard input	standard output
3 5	3
3 1	0
4883 5989	956662807

Problem G. Game with Ninja

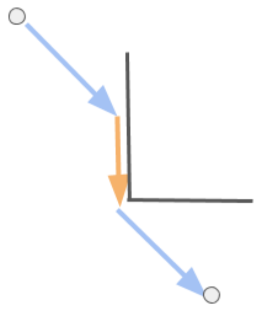
Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In the new game "Follow the Way of Ninja" the player manipulates the ninja on the two-dimensional map which is represented by a polygon without self-intersection consisting of only sides parallel to either the x-axis or the y-axis.

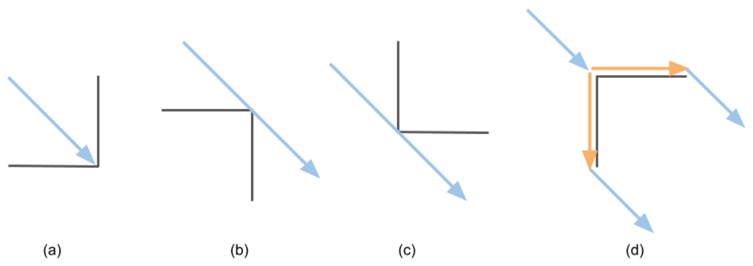
Ninja can move in 8 directions: upward, downward, leftward, rightward and in 4 diagonal directions. As long as ninja is ordered to run in some direction, it runs as long as it is possible, i.e. ninja stops only in two cases:

- ninja got the new order; in this case the direction is immediately changed;
- ninja's movement is blocked in `all` directions of movement, i.e. if ninja moves diagonally and reaches the wall, he does not stop but continues moving along this wall until it presents, then if the wall ends and nothing is blocking the previous (diagonal) direction, ninja continues to move at this direction.

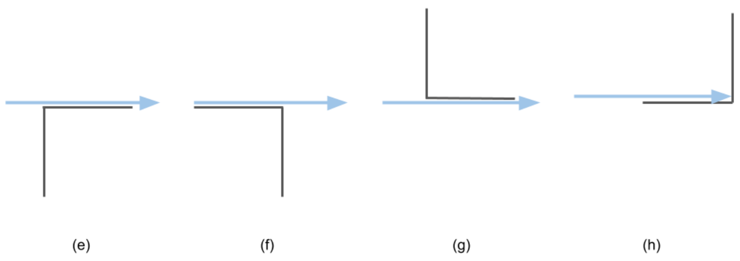
Consider some examples. If your ninja tries to move in a direction perpendicular to the wall, he can not move any further and stop. If a wall parallel to the y axis is hit by an diagonal direction consisting of the x axis positive direction and the y axis negative direction as shown in the figure below, after colliding, ninja proceeds along the wall along the y- axis negative direction, where the wall disappears. Then ninja will continue movement in the diagonal direction composed by movement along the x- axis positive direction and the y- axis negative direction.



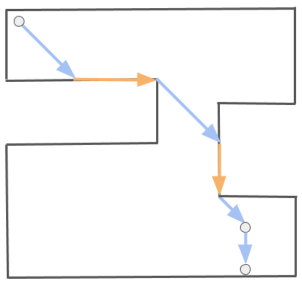
Below are examples of ninja's behavior while hitting a corner while moving diagonally. If the wall is enclosed in both directions as shown in (a), ninja stops there, and can not move without changing direction. As shown in (b) and (c), if ninja can go straight through the corner, it continues to move automatically in an diagonal direction. If ninja can proceed in both directions as shown in (d), it can choose the way you like (and this will not be counted as the order). Note that on the figure, since the wall is hidden by the arrow representing movement, it is shown slightly toward the wall, but this also shows that it actually moves on the wall.



In addition, the behaviors when hitting a corner during vertical and lateral movement are as follows. In cases like (e), (f), (g), continue to move automatically as it is. If ninja strikes the wall perpendicularly as shown in (h), it stops there and can not move without changing direction. Similarly, the arrow indicating movement is slightly shifted towards the inside from the wall, but in fact it shows that it moves on the wall.



You are given the starting point of the ninja and the target point, both points are inside of polygon. Calculate mininal number of orders you need to move a ninja from start to finish. Fir example, the minimum number of orders in the figure below is 2 as shown in the figure. This corresponds to the third sample input.



Input

The input consists of multiple data sets. The maximum number of data sets is 100.

The first row of the data set consists of one integer N ($4 \leq N \leq 100$) representing the number of vertices of the polygon representing the map. The second line consists of four integers s_x , s_y, g_x, g_y ($-10^4 \leq sx, sy, gx, gy \leq 10^4$), the coordinates of the starting point are (s_x, s_y), the coordinates of the finish are (g_x, g_y) . In the following N rows, vertices of the polygon are listed in counterclockwise order. The i-th row consists of two integers x_i, y_i ($-10^4 \leq x_i, y_i \leq 10^4$), the coordinates of the i-th vertex. Additionally, the following restrictions are satisfied:

- All edges are parallel to either x or y axis.
- The given polygon has no self-intersection. That is, each side is not intersecting other side outside the end points.
- It is guaranteed that both (s_x, s_y) and (g_x, g_y) are strictly inside the polygon and are different.

The input is terminated with the test case with $N = 0$, which shouldn't be processed.

Output

For each data set, output the minimum number of command inputs required to arrive from the start point to the goal point on one line.

Examples

standard input

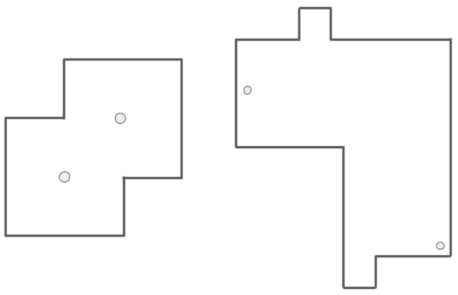
```
8
1 1 2 2
0 2
0 0
2 0
2 1
3 1
3 3
1 3
1 2
12
-9 5 9 -9
0 0
0 -13
3 -13
3 -10
10 -10
10 10
-1 10
-1 13
-4 13
-4 10
-10 10
-10 0
12
3 57 53 2
0 0
64 0
64 18
47 18
47 39
64 39
64 60
0 60
0 44
33 44
33 30
0 30
0
```

standard output

```
1
1
2
```

Note

Polygons for test cases 1 and 2:



(1) (2)

Problem H. How Much Bits?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given a value k and a number of bits b, calculate the total number of 1-bits in the binary representations of all multiples of k that are between 0 and $2^b - 1$ (inclusive).

Input

Input will consist of a single line containing two space-separated integers k ($1 \leq k \leq 1,000$) and b ($1 \leq b \leq 128$), where k and b are as described above.

Output

Output a single integer, which is the total number of 1-bits in the binary representations of all multiples of k that are between 0 and $2^b - 1$ (inclusive). Since this number may be very large, output it modulo $10^9 + 9$.

Examples

standard input	standard output
1 4	32
10 5	8
100 7	3
3 28	252698795
11 128	856188165
1 26	872415232
876 128	530649653

Note

Consider the second sample: $k = 10$ and $b = 5$.

$2^5 - 1 = 31$. All the multiples of 10 between 0 and 31 are: 10, 20 and 30.

$10 = 01010_2$ (2 1-bits)

$20 = 10100_2$ (2 1-bits)

$30 = 11110_2$ (4 1-bits)

That's a total of 2+2+4=8 1-bits.

Problem I. Inversions

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a sequence of n integers, all of them between 1 and k (inclusive). Some of the integers are missing, and are replaced with 0's.

An inversion is a pair of values a_i and a_j in the sequence, where $i < j$, but $a_i > a_j$. What is the maximum number of inversions possible if the missing integers are all between 1 and k inclusive?

Input

Input will start with a line with two space-separated integers n ($1 \le n \le 2 \cdot 10^5$) and k ($1 \le k \le 100$), where n is the length of the sequence and k is the maximum value of elements of the sequence.

Each of the next n lines will contain a single integer x ($0 \le x \le k$). This is the sequence, in order, with 0's representing the missing values.

Output

Output a single integer, which is the maximum number of inversions possible.

Examples

standard input	standard output

6 9
0
8
4
3
0
0
	15

10 9
5
2
9
0
7
4
8
7
0
0
	28

10 9
7
4
0
0
8
5
0
0
3
1
	36

Problem J. Japanese Lock

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The lock on the door of the very secret agency consists of N cylinders, each with M sequential integers between 0 and M on it. Initially at the upper side of the lock are N zeroes. The lock opens when sequence T of N integers between 0 and M is at the upper side.

You may rotate the cylinders **only in pairs**. Each time you have to choose two arbitrary distinct cylinders and rotate them sychronously. You may rotate both of the selected cylinders in the forward direction, or both in the backward direction. Rotation one step in the forward direction replaces the value A with $(A+1) mod M$, one step in the backward direction replaces it with $(A-1) mod M$.

Find the minimum number of steps, required to open the lock, or -1 if it is impossible to open it.

Input

First line of the input contains two integers N and M ($2 \le N \le 300$, $2 \le M \le 10^5$). Second line contains the key T — N integers T_1, …, T_N ($0 \le T_i \le M-1$).

Output

Output the minimum number of rotation steps needed to open the lock or -1 if it is impossible to open it.

Examples

standard input	standard output
2 3 0 1	-1
3 7 0 4 4	3
3 6 2 4 2	4

15 56
19 32 40 34 4 10 30 15 4 18 52 28 18
8 4
	110

Problem K. Knockout

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The solitaire game Knockout is played as follows. The digits from 1 to 9 are written down in ascending order. In each turn, you throw a pair of six-sided dice; you sum the dice, and cross out some set of digits, of your choice, that sum to the same total. If you cannot, the game ends and your score is the remaining digits, taken as a single number. Otherwise, you throw the dice again and continue.

This game can be played to either minimize or maximize your score. Given a position of the game (what digits remain) and a roll of the dice, determine which digits you should remove and what your expected score would be for both versions of the game, assuming you make the best moves possible for whichever version youre playing for the remainder of the game. The expected score is the sum of all possible scores weighted by their probabilities (presuming optimum play).

Input

Input will consist of a single line containing a string of digits d ($1 \le |d| \le 9$) and two integers a and b ($1 \le a, b \le 6$), all separated by spaces. The string of digits d will contain a subset of the digits '`1`'..'`9`' in ascending order, with no digit appearing more than once. This is the current state of the game. The integers a and b represent your current throw of the dice.

Output

Output two lines, each with two parts. First, output the digits that you eliminate with your throw of the dice, as a string of digits in ascending order. If you cannot eliminate any digits, output -1. Then, output the expected score with absolute error 10^{-5} or better. Output a space between the parts.

The first line represents the best result when minimizing your score and the second line represents the best result when maximizing your score. Note that it is impossible for two different combinations of digits to yield the same expected score.

Examples

standard input	standard output
`1345 1 1`	`-1 1345.00000` `-1 1345.00000`

`12349 3 1`	`13 151.70370`
	`4 401.24546`

Problem L. Lines Processing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a two-dimensional array of pairs of integers $A_{x,y}$, infinite in all four directions. Initially each element has its coordinates as its value, i. e. $A_{x,y} = (x,\ y)$. You need to process Q queries numbered from 1 to Q. Each query, say i-th, can be of one of the following types:

1) Insert a (infinite) row at $y = k$, shifting rows with $y \geq k$. The elements of the inserted line have values $(x,\ i)$. More formally, let B denote array A after processing the query, then adding a row at $y = k$ means that (note that i is a number of the current query)

$$B_{x,y} = \begin{cases} A_{x,y}, & y < k \\ (x, i), & y = k \\ A_{x,y-1}, & y > k \end{cases}$$

2) Insert a column at $x = k$. Defined similarly to the previous query type:

$$B_{x,y} = \begin{cases} A_{x,y}, & x < k \\ (i, y), & x = k \\ A_{x-1,y}, & x > k \end{cases}$$

3) Get the value of $A_{x,y}$ for given x and y.

Input

The first line of the input contains integer Q ($1 \leq Q \leq 10^5$), the number of queries. Q queries follow. Let (p, q) be the answer to the latest query of type 3, or both zero if there was no type 3 query yet. Each query has one of the forms:

1) $1\ k_0$ — insert a row at $k = k_0 + p + q$.

2) $2\ k_0$ — insert a column at $k = k_0 + p + q$.

3) $3\ x_0\ y_0$ — get $A_{x,y}$, where $x = x_0 + p + q$, $y = y_0 + p - q$ (note the "$-$" sign). Remember that this query also assigns its result to $(p,\ q)$.

For each query: $-10^7 \leq k, x, y \leq 10^7$.

Output

For each query of the third type output a line containing two integer numbers separated by a space, the value of $A_{x,y}$ at the moment of the query.

Examples

standard input	standard output

5
3 1 2
3 1 2
2 -6
1 -3
3 -6 -1
	1 2
	4 1
	-1 4

Problem M. Mobilization

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are playing a strategy game in which you are required to mobilize an army. The army consists of different types of troops, each of which has a cost, health, and potency. You can acquire any combination of the troop types, even fractional, such that the total cost is no more than the amount of money you have to spend.

The strength of the army is equal to its total health value multiplied by its total potency (i.e. $(\sum a_i \cdot h_i) \times (\sum a_i \cdot p_i)$, where h_i is the health of troop type i, p_i is the potency of troop type i, and a_i is the amount acquired of troop type i).

What is the greatest strength you can achieve given the troops available and the money in your coffers? You may assume that there will always be sufficient troops to buy as many as you can afford.

Input

Input will begin with a line with two space-separated integers n ($1 \le n \le 3 \cdot 10^4$) and m ($0 \le m \le 10^5$), where n is the number of troop types and m is the total amount of money you have to spend.

Each of the next n lines will hold three values separated by spaces, representing a type of troop: c, h, p, where c ($1 \le c \le 10^5$) is an integer, which is the cost of that type of troop, h ($0.0 \le h \le 1.0$) is a real number, which is the health of that type of troop, and p ($0.0 \le p \le 1.0$) is a real number, which is the potency of that type of troop.

Output

Output a single real number, which is the maximum strength you can achieve with your monetary resources. Output this number with absolute error 10^{-2} or less. to exactly 2 decimal places.

Example

standard input	standard output

4 100000
300 1 0.02
500 0.2 1
250 0.3 0.1
1000 1 0.1
	19436.05

2 100
1 0.1 1
1 1 0.1
	3025.00