Problem A
Flipping Coins

Here's a jolly and simple game: line up a row of N identical coins, all with the heads facing down onto the table and the tails upwards, and for exactly K times take one of the coins, toss it into the air, and replace it as it lands either heads-up or heads-down. You may keep all of the coins that are face-up by the end.

Being, as we established last year, a ruthless capitalist, you have resolved to play optimally to win as many coins as you can. Across all possible combinations of strategies and results, what is the maximum expected (mean average) amount you can win by playing optimally?

Input

- One line containing two space-separated integers:
  - N ($1 \leq N \leq 400$), the number of coins at your mercy;
  - K ($1 \leq K \leq 400$), the number of flips you must perform.

Output

Output the expected number of heads you could have at the end, as a real number. The output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
`2 1`	`0.5`

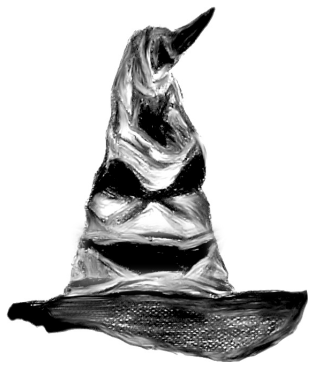
Sample Input 2	Sample Output 2
`2 2`	`1`

Sample Input 3	Sample Output 3
`2 3`	`1.25`

Sample Input 4	Sample Output 4
`6 10`	`4.63476563`

Sample Input 5	Sample Output 5
`6 300`	`5.5`

Problem B
Deranging Hat



In the wizarding world of security, there are two kinds of researcher: the idealist *arranging hat* and the mercenary *deranging hat*.

As we learned last year, an *arranging hat* carefully sorts out any list of letters given to it into ascending order. However, a *deranging hat* performs the exact opposite function: putting a sorted string of letters back into its original order.

The tool of choice for today's discerning headwear is a sorting network: a sequence of instructions represented by a list of pairs of numbers A_i and B_i, meaning that if at step i the A-th item in the string is not already smaller than the B-th item, they should be swapped immediately.

Given a specific word W, output a sorting network that the deranging hat can use to form the word from its original sorted letters.

Input

One line containing one string of lowercase Latin letters ('a'-'z'), S, containing at most 1000 characters.

Output

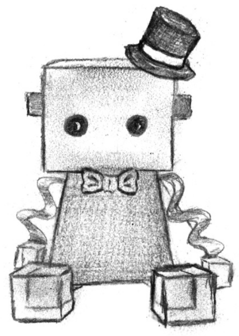
Output at most 10000 lines, each containing two integers A_i and B_i ($1 \leq A_i, B_i \leq |S|$) giving the i-th operation to perform.

Sample Input 1	Sample Output 1
`bab`	`2 1`

Sample Input 2	Sample Output 2

`dude`	`4 3`
	`3 2`

Problem C
GentleBots



Rainforest Inc. is opening a large new automated warehouse in the far Northern reaches of the UK—some place they call "Walthamstow".

The robotic worker drones inside will operate in not just one or two, but three dimensions, and can move in any one of the 6 cardinal dimensions in steps of 1 metre at a time. For example, a robot looking to move from position (X_1, Y_1, Z_1) to position (X_2, Y_2, Z_2) will, assuming no obstacles, need to take $(|X_2 - X_1| + |Y_2 - Y_1| + |Z_2 - Z_1|)$ steps in total.

Since this warehouse is in Britain, and because every stereotype is true, the robotic denizens are all impeccably polite. When two robots travelling in opposite directions meet, they wordlessly negotiate for one of the robots to step aside somehow so the other can pass.

Multiple robots cannot occupy the same integer co-ordinates, and no two robots can swap positions in one move. All moves are instantaneous.

We have prepared a test run of the warehouse with just two machines installed. Write a program to pass this test.

Input

Two lines, one for each robot, each containing six space-separated integers $(X_0 Y_0 Z_0)$ and $(X_\infty Y_\infty Z_\infty)$, the intended start and end locations of a robot respectively $(-1000 \leq X, Y, Z \leq 1000)$.

The robots will start in different places from each other, and will also end in different places from each other.

Output

Output up to 7000 lines, giving one possible list of locations of the robots over time. The position of both robots at time T must be given as bracketed space-separated (X, Y, Z) co-ordinate tuples on line T.

Co-ordinates must not exceed an absolute value of 10^6.

Sample Input 1

```
0 0 0 2 2 2
1 1 2 0 0 0
```

Sample Output 1

```
(0 0 0) (1 1 2)
(1 0 0) (1 1 1)
(1 1 0) (0 1 1)
(1 1 1) (0 1 0)
(1 1 2) (0 0 0)
(1 2 2) (0 0 0)
(2 2 2) (0 0 0)
```

Sample Input 2

```
-2 0 0 1 0 0
3 0 0 -1 0 0
```

Sample Output 2

```
(-2 0 0) (3 0 0)
(-1 0 0) (2 0 0)
(0 0 0) (1 0 0)
(1 0 0) (1 0 -1)
(1 0 0) (0 0 -1)
(1 0 0) (-1 0 -1)
(1 0 0) (-1 0 0)
```

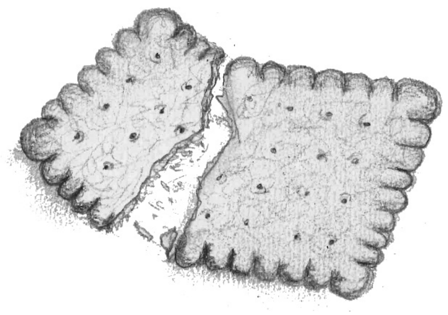
Sample Input 3

```
0 0 0 1 0 0
1 0 0 0 0 0
```

Sample Output 3

```
(0 0 0) (1 0 0)
(0 1 0) (0 0 0)
(1 1 0) (0 0 0)
(1 0 0) (0 0 0)
```

Problem D
Breaking Biscuits



This year, Walter's workplace resolved to try something radically different: they're going to change the weekly order of biscuits for the break room to a whole other brand.

Biscuits come in many shapes and sizes, but the particular brand they settled on has two special qualities:

- It is completely planar (two-dimensional);
- It is perfectly polygon-shaped.

However, disaster struck immediately: the available mugs in the break room are too narrow for Walter to be able to dunk these new biscuits into, no matter what combination of rotations along the three axes he tries.

There is only one thing for it: Walter will need to order another mug.

Before taking this drastic step, it is vital to know how wide the diameter of this mug will need to be in order to succesfully accommodate a (possibly rotated) biscuit of the new brand.

Input

- One line containing an integer N ($3 \leq N \leq 100$), the number of vertices in the biscuit.
- Each of the following N lines contains two space-separated integers X_i and Y_i ($-10^5 \leq X_i, Y_i \leq 10^5$), the coordinates of the i-th vertex.

Vertices are always given in anti-clockwise order. Also, as anyone can tell you, biscuits never self-intersect and always have positive area.

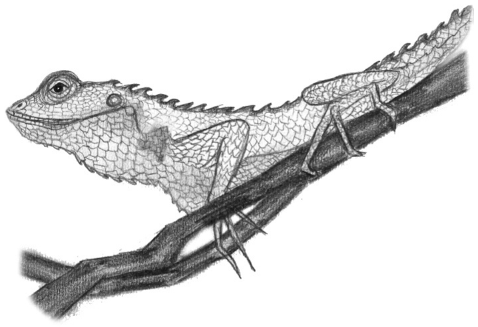
Output

Output the minimum possible diameter of the new mug, in order that it can fit the new kind of biscuit fully inside in at least one orientation. The output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1

4
0 0
5 0
5 2
0 2
	2.0

Problem E
Lounge Lizards



Monitor lizards are a kind of reptile known mainly for their cold-bloodedness and addiction to computer screens. Due to their love for digital displays, these scaly creatures spend most of their time at home glued to a small television in the lounge.

Conflict has arisen at one particular reptile house. The audience here has grown so large that not everyone will be able to see the screen at once any more; specifically, a lizard will only be able to see enough if it is strictly taller than all of the lizards sitting exactly along the straight line from itself to the television.

Monitor lizards aren't particularly picky about the actual contents of the screen or being able to see it obliquely (or even from the front)—they just want to be able to keep an eye on it.

The lizards don't want to move, however. It's possible to chase a monitor lizard away in order for the ones behind it to see, or leave it alone, but re-homing somewhere else in the room is unthinkable.

Assuming lizards are removed optimally, how many at most can remain and still see the screen?

Input

- one line containing the space-separated integers T_X and T_Y ($-10^6 \leq T_X, T_Y \leq 10^6$), the co-ordinates of the television.
- one line containing the integer N ($1 \leq N \leq 10^6$), the number of lizards.
- N further lines, each containing three space-separated integers $X_i Y_i H_i$ ($-10^6 \leq X, Y \leq 10^6; 1 \leq H \leq 10^6$), the co-ordinates and height respectively of one lizard.

The co-ordinates of all televisions and lizards will be distinct.

Output

Output the maximum number of lizards that can stay and watch television at once.

Sample Input 1	Sample Output 1

50 50
2
60 50 1
65 50 2
	2

Sample Input 2

```
50 50
3
60 55 1
70 60 1
40 45 1
```

Sample Output 2

```
2
```

Sample Input 3

```
-100 0
6
-99 1 2
-98 2 4
-97 3 3
-96 4 4
0 100 3
100 0 7
```

Sample Output 3

```
4
```

Problem F
Craters

General Warren Pierce has a bit of a problem. He's in charge of a new type of drone-delivered explosive and they've been testing it out in the Nevada desert, far enough from any population center to avoid civilian casualties and prying eyes. Unfortunately word has gotten out about these experiments and now there's the possibility of careless on-lookers, nefarious spies, or even worse — nosy reporters! To keep them away from the testing area, Warren wants to erect a single fence surrounding all of the circular craters produced by the explosions. However, due to various funding cuts (to support tax cuts for the you-know-who) he can't just put up miles and miles of fencing like in the good old days. He figures that if he can keep people at least 10 yards away from any crater he'll be okay, but he's unsure of how much fencing to request. Given the locations and sizes of the craters, can you help the General determine the minimum amount of fencing he needs? An example with three craters (specified in Sample Input 1) is shown below.

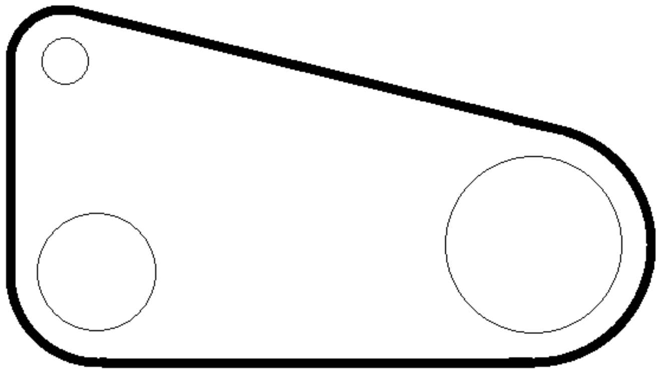


Figure B.1: Three craters with a fence around them.

Input

The first line of input contains a single positive integer n $(n \leq 200)$, the number of craters. After this are n lines specifying the location and radius of each crater. Each of these lines contains 3 integers x y r, where x and y specify the location of a crater $(|x, y| \leq 10\,000)$ and r is its radius $(0 < r \leq 5\,000)$. All units are in yards.

Output

Display the minimum amount of fencing (in yards) needed to cordon off the craters, with an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1

3
0 0 100
-60 200 40
350 50 150
	1715.91229929