

Sledding down the mountain

Writer: Victor Lecomte

Time limit: 12s Memory limit: 256 MB

You are on vacation in the mountains, and are going on a crazy sled ride through hilly terrain. You can easily go down a slope and you can also travel while staying at constant height, but you can't go up, that is, your z -coordinate is nonincreasing.

The surface of the mountain is formed of triangle faces linked together at their edges. There are no vertical slopes and no overhangs. You should always stay on those triangle faces, but you are allowed to move on their edges and to move from one triangle to another *if they share an edge*.

Given a starting point on the mountain, what is the total surface occupied by the points that you could reach on the mountain without going up?

Input

The first line of the input contains one integer n , the number of triangles in the mountain. Then follow n groups of 3 lines. The i^{th} group contains three 3D points $P_{i,1}, P_{i,2}, P_{i,3}$, the vertices of the triangle, each one one line. Each vertex is described by 3 space-separated integers $x_{i,j}, y_{i,j}, z_{i,j}$, its coordinates. The z -coordinate represents the height, while the x - and y -coordinates represent the position of the point on a map of the mountain.

Your starting point is $P_{1,1}$.

Output

Output a floating-point value on a single line: the total area of the mountain that you can reach on your sled without going up. The answer should be accurate to an absolute or relative error of 10^{-6} .

Limits

- $1 \leq n \leq 10^5$;
- $-10^5 \leq x_{i,j}, y_{i,j}, z_{i,j} \leq 10^5$;
- the vertices of a triangle are given in counter-clockwise order, that is, the z -coordinate of $\overrightarrow{P_{i,1}P_{i,2}} \times \overrightarrow{P_{i,1}P_{i,3}}$ is positive;
- the mountain is connected, that is, you can go from any triangle to any other triangle by passing through common edges;

Sledding down the mountain

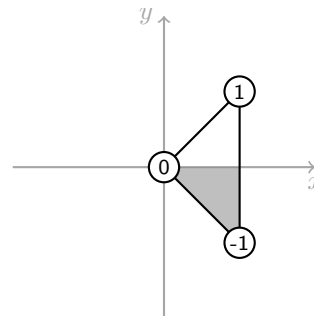
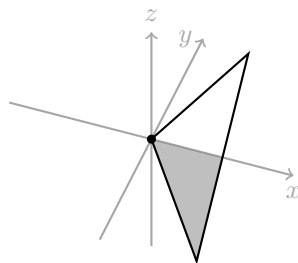
- if two triangles touch, they share either an entire edge, or one vertex;
- no point in a triangle is directly above or below another triangle.

Example 1

stdin	stdout
1 0 0 0 1 -1 -1 1 1 1	0.707106781187

This mountain has only one triangle. You start at point $(0,0,0)$ and you can ride your sled to the whole lower half of the triangle, that is, you can ride to any point on the triangle of vertices $(0,0,0)$, $(1,-1,-1)$ and $(1,0,0)$ (in gray). This is a right triangle with sides 1 and $\sqrt{2}$, so its area is $\frac{\sqrt{2}}{2}$.

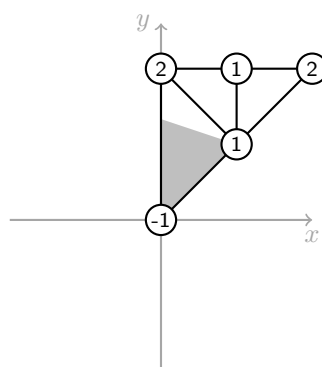
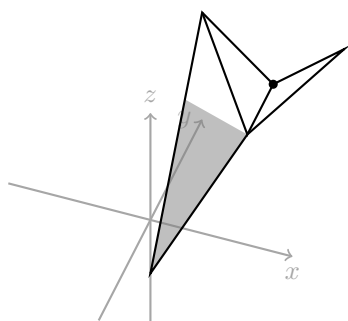
Below on the left is a 3D view of the mountain, and on the right a view from above, with the height marked at every vertex. The starting point is marked with a dot in the 3D view, and the reachable region is highlighted.



Example 2

stdin	stdout
<pre> 3 1 2 1 0 2 2 1 1 1 1 1 1 2 2 2 1 2 1 0 0 -1 1 1 1 0 2 2 </pre>	<pre> 1.247219128925 </pre>

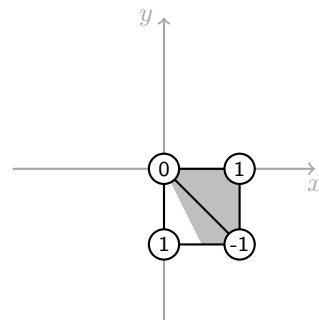
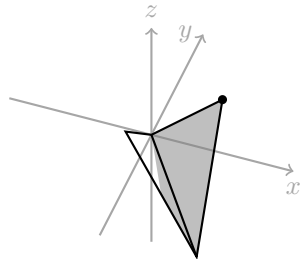
This mountain has three triangles. Triangle 1 has the edge from $(1, 2, 1)$ to $(1, 1, 1)$ in common with triangle 2, and the edge from $(1, 1, 1)$ to $(0, 2, 2)$ in common with triangle 3. You start at point $(1, 2, 1)$, at height 1. You cannot climb up triangles 1 or 2, so the area that you cover on them is zero. But you can travel on the edge from $(1, 2, 1)$ to $(1, 1, 1)$ at constant height to get to triangle 3. From there, you can reach $\frac{2}{3}$ of the area of triangle 3, which gives the answer $\frac{2}{3} \times \frac{\sqrt{14}}{2}$.



Example 3

```
      stdin
2
1 0 1
0 0 0
1 -1 -1
1 -1 -1
0 0 0
0 -1 1
```

```
      stdout
1.837117307087
```



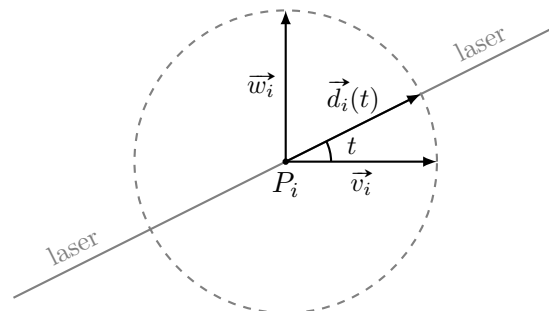
Pen spinning accident

Writer: Victor Lecomte

Time limit: 1 s Memory limit: 256 MB

You and your friend are huge pen spinning experts, and since it was getting too easy you decided to take it to the next level by mounting deadly lasers on both ends of your pens. While you don't worry much about accidentally hurting people, there is one safety guideline that you must respect: much like in a famous '80s movie, you should never cross the laser beams, because it would be... bad. So you would like to know when the beams are going to cross, if ever, so that you can safely stop before it happens.

In the rest of the statement, you will be called "person 1" and your friend "person 2". Person i is currently spinning their pen around point P_i in space at a constant rate of 1 radian per second. At time t their pen is pointing in direction $\vec{d}_i(t) = (\cos t)\vec{v}_i + (\sin t)\vec{w}_i$, that is, the laser beams on both ends run along the line containing points P_i and $P_i + \vec{d}_i(t)$. You can consider that the length of the pen is negligibly small, so that the whole line is covered by the laser beams.



Input

The output is made of 2 groups of 3 lines, separated by a blank line. The i th group contains the values $P_i, \vec{v}_i, \vec{w}_i$ on separate lines. Each of them consists of three space-separated integers: their x -, y - and z -coordinates.

Output

The output is a single line. Print the earliest time $t > 0$ at which the beams cross, if it exists. Otherwise, print the word **never**. The time should be accurate to an absolute error of 10^{-7} .

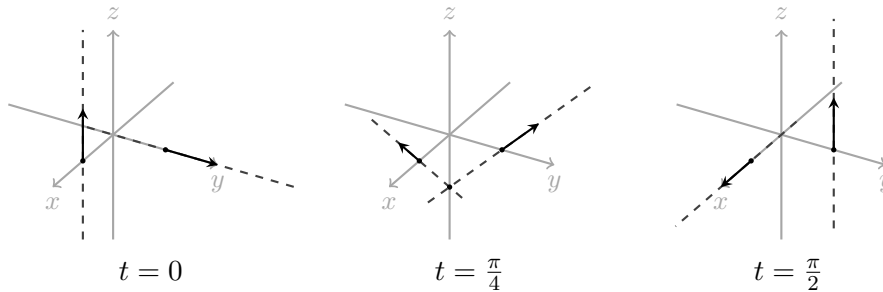
Limits

- all coordinates are integers in $[-100, 100]$;
- $P_1 \neq P_2$ (you are at distinct locations);
- $\vec{v}_i \cdot \vec{w}_i = 0$ and $\|\vec{v}_i\| = \|\vec{w}_i\| \neq 0$ (so direction $\vec{d}_i(t)$ rotates on a circle);
- $\vec{d}_1(t) \times \vec{d}_2(t) \neq \vec{0}$ for all t (the beams are never parallel);
- the laser beams are not touching at time 0;
- you will never shoot each other with your lasers, that is, P_1 is not on the line containing P_2 and $P_2 + \vec{d}_2(t)$ for any t , and vice versa.

Example 1

stdin	stdout
<pre> 1 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 </pre>	<pre> 0.785398163397 </pre>

In this example, the starting directions of the pens are $(0, 0, 1)$ and $(0, 1, 0)$, so the lines formed by their lasers are at $x = 1, y = 0$ and $x = 0, z = 0$ respectively. After $\pi/4$ seconds, pen 1 points in direction $\frac{1}{\sqrt{2}}(1, 0, 1)$ while pen 2 points in direction $\frac{1}{\sqrt{2}}(0, 1, 1)$, so that both lines contain point $(0, 0, -1)$. This is the first time they touch.



Example 2

stdin

1 0 0

0 0 1

1 0 0

0 1 0

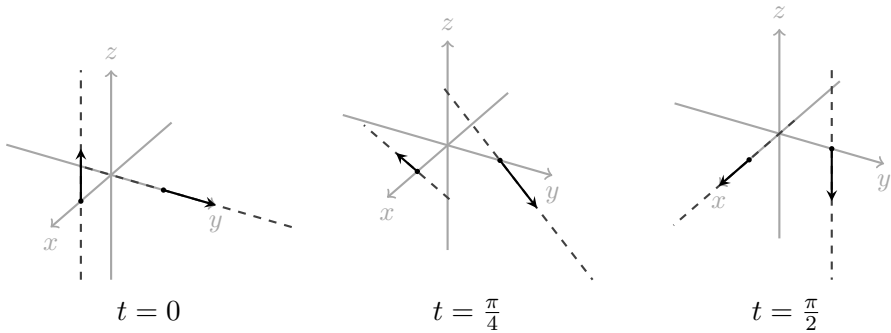
0 1 0

0 0 -1

stdout

never

This example is identical to the first, except that the second pen rotates in the opposite direction.



Example 3

stdin

0 0 0

1 0 0

0 1 0

0 0 1

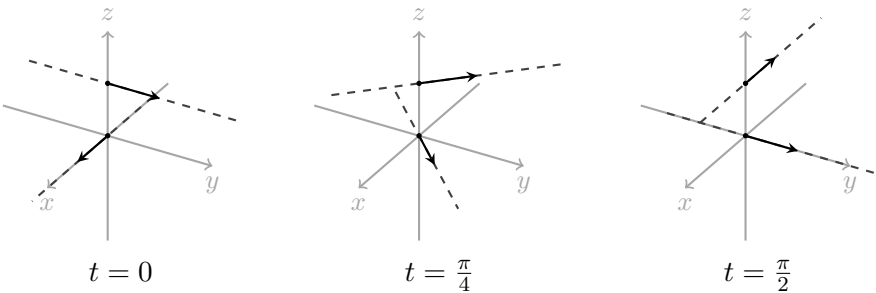
0 1 0

-1 0 0

stdout

never

In this example, the lasers of the first pen only reach points for which $z = 0$, while the lasers of the second pen only reach points for which $z = 1$, so they clearly never touch.



Example 4

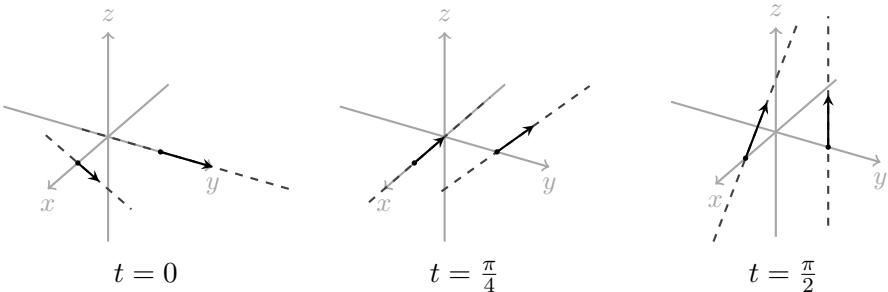
stdin

```
1 0 0
-1 0 -1
-1 0 1

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

stdout

0.392699081699



Example 5

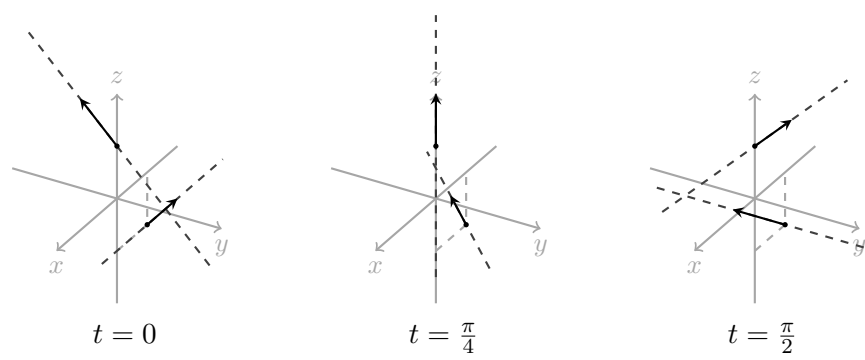
stdin

```
0 0 1
0 -1 1
0 1 1

-1 0 -1
-1 0 0
0 -1 0
```

stdout

0.511669644114



Example 6

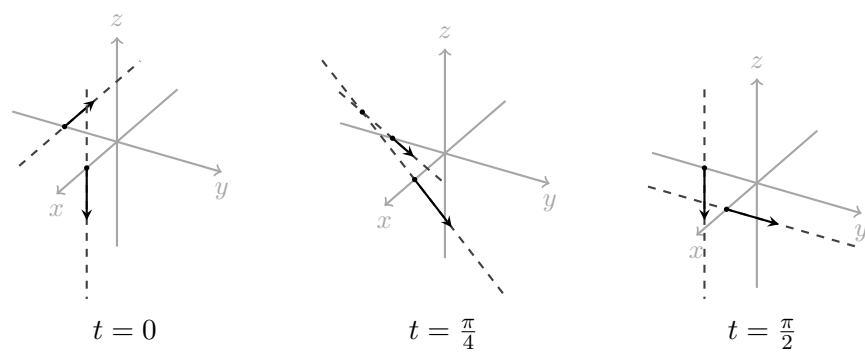
stdin

```
0 -1 0
-1 0 0
0 0 -1

1 0 0
0 0 -1
0 1 0
```

stdout

0.785398163397



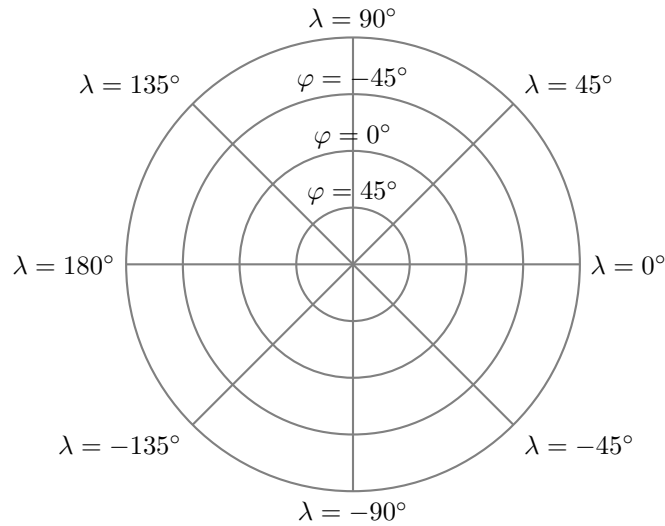
Polar map

Writer: Victor Lecomte

Time limit: 8 s Memory limit: 256 MB

You live on the North Pole, and are frustrated that your house is located on the edge (or even outside) of most world maps. So you have decided to create your own map, with two criteria: the North Pole should be in the center, and distances from it should be conserved. The *azimuthal equidistant projection*¹ is perfect for this.

In the azimuthal equidistant projection, the map is a circle of radius 1, and a point of latitude φ and longitude λ (given in degrees) will be represented at position $(d \cos \lambda, d \sin \lambda)$, where $d = \frac{90^\circ - \varphi}{180^\circ}$.



Now, consider a country, whose border is determined by some border points, where each border point is linked to the next by the shortest path along the Earth's surface (the *great-circle distance*²). What is the area occupied by that country on your map?

Input

The first line of the input contains one integer n , the number of border points of the country. Then follow n lines, the i^{th} of which contains two integers φ_i and λ_i , the latitude and longitude of the border point, in degrees. The points are given in counter-clockwise order, that is, when going from point i to point $i + 1$, the inside of the country is on the left side.

¹See https://en.wikipedia.org/wiki/Azimuthal_equidistant_projection for a more in-depth introduction.

²See https://en.wikipedia.org/wiki/Great-circle_distance for a more in-depth introduction.

Output

Output a floating-point value on a single line: the area occupied by the country on your map. The answer should be accurate to an absolute error of 10^{-7} .

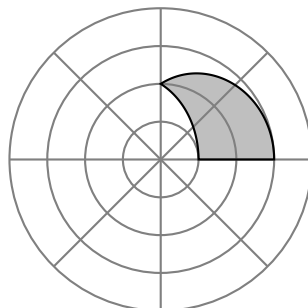
Limits

- $3 \leq n \leq 10^3$;
- $-85 \leq \varphi_i \leq 85$, $-180 < \lambda_i \leq 180$;
- the Earth is a perfect sphere (sorry, flat earthers);
- the country does not self-intersect;
- no two consecutive points on the border of the country are exactly opposite each other;
- the country contains neither the North Pole, nor the South Pole;
- no border of the country will come closer than 2×10^{-4} to the center of the map (North Pole) or to the border of the map (South Pole).

Example 1

stdin	stdout
<pre>3 45 0 -45 0 0 90</pre>	<pre>0.269064434316</pre>

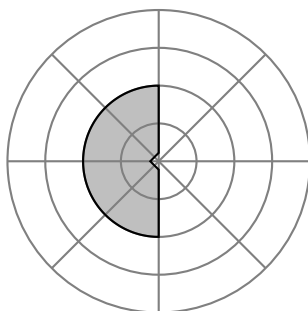
The border points of the country are located at $(0.25, 0)$, $(0.75, 0)$ and $(0, 0.5)$ on the map. However, the area is not $1/8$ because the borders of the country are not straight lines on the map.



Example 2

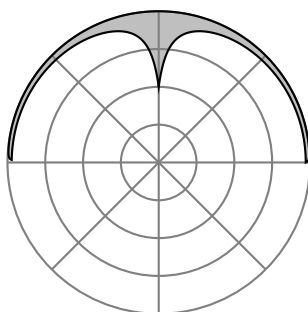
stdin	stdout
6 0 90 0 180 0 -90 80 -90 80 180 80 90	0.389591701512

Between the first three points, the border runs along the equator, which is a circular arc on the map. The area is a bit less than $\pi/8$.



Example 3

stdin	stdout
3 -85 0 -85 179 0 90	0.207400294880



Gold paint optimization

Writer: Victor Lecomte

Time limit: 10 s Memory limit: 256 MB

You are working on a modern art piece consisting of n cubes of side 1 m in space. The cubes are currently unpainted, and that's a bit boring, so you have decided to buy some gold paint to decorate them. Since the gold paint is made of real gold, it's very expensive, so you only have l liters. One liter can paint a surface of 1 m^2 . You are allowed to distribute the paint completely as you wish, painting only some of the cubes, only some of the faces of a given cube, and even only part of a given face.

The visitors will come to see your work at point $(0, 0, 0)$, so you would like to optimize your use of the paint so that the part of their field of view that is covered by gold paint is as large as possible, that is, the total *solid angle* subtended at $(0, 0, 0)$ by the visible painted parts of the cubes is maximized.

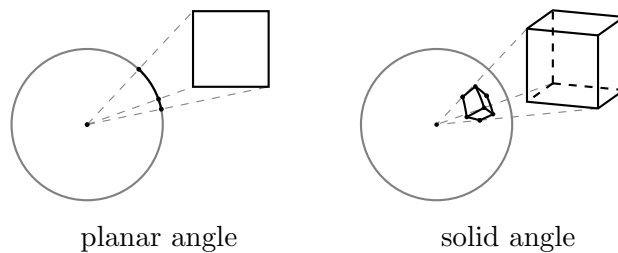
Paint does not count if it is placed on faces that are not visible from the origin because they are on the wrong side of the cube.

To achieve this goal, how much paint should you use on each of the cubes?

Solid angle

Just like the planar angle subtended by an object is the length of the object once it is projected onto a unit circle around the point, the solid angle subtended by an object is the area of the object once it is projected onto a unit sphere around the point.

See https://en.wikipedia.org/wiki/Solid_angle for a complete definition.



Input

The first line of the input contains two integers n and l , the number of cubes and the quantity of paint in liters. Then follow n lines, the i^{th} of which contains three integers x_i, y_i, z_i , meaning that there is an axis-aligned unit cube at $[x_i, x_i + 1] \times [y_i, y_i + 1] \times [z_i, z_i + 1]$.

Output

The output consists of n lines, the i^{th} of which contains a floating-point value l_i , the number of liters of paint that should be used on cube i . The numbers should be accurate up to a relative or absolute error of 10^{-4} . It can be proven that the solution is unique under the constraints of this problem.

Limits

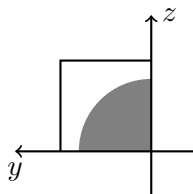
- $1 \leq n \leq 10^4$;
- $0 \leq l < 3n$;
- $0 \leq x_i, y_i, z_i < 100$, which means that the faces visible from the origin are in planes $x = x_i$ if $x_i \neq 0$, $y = y_i$ if $y_i \neq 0$ and $z = z_i$ if $z_i \neq 0$;
- no cube touches the origin;
- no cube hides any part of another cube, that is, no ray from the origin touches more than one cube;
- you don't have enough paint to cover all visible faces, that is, if you had more paint, you could strictly increase the solid angle subtended at the origin by the visible painted parts.

Example 1

stdin	stdout
2 1	0.50000000
2 0 0	0.50000000
0 2 0	

You have 1 liter to paint two cubes at $[2, 3] \times [0, 1] \times [0, 1]$ and $[0, 1] \times [2, 3] \times [0, 1]$. They are at equal distances from $(0, 0, 0)$ and have only one visible face from there, at $x = 2$ and $y = 2$, respectively. It is optimal to use half of the paint on each face.

Here is what the visible face of the first cube looks like from the origin, with the painted part highlighted:



Example 2

stdin	stdout
2 2	1.00000000
2 0 0	1.00000000
10 10 10	

You have 2 liters to paint two cubes at $[2, 3] \times [0, 1] \times [0, 1]$ (one visible face) and $[10, 11] \times [10, 11] \times [10, 11]$ (three visible faces). It is optimal to paint the visible face of the first cube completely, then to use the remaining liter to paint part of the three faces of the second cube.

Example 3

stdin	stdout
2 2	0.98985432
2 0 0	1.01014568
0 1 1	