Problem A: The Cubical Walk

1 Description

Once upon a time there lived a cubical man. His name was Mikhail Kubachev. He lived in a cubical world on a cubical planet. This planet was, in fact, in the form of parallelepiped. And the man had the parallelepiped body, too. Nothing is perfect, so his body had internationally famous marked spot. Mr. Kubachev was an adept of the healthy lifestyle and he used to walk on his planet each evening.

Imagine a man staying on the surface of the planet and having a parallelepiped body. He has the direction "forward" as the direction in which he is intended to walk, he has the direction "up" and "down", "left", "right" and "back". Directions "up" and "down" are determined by the planet's face as vectors which are perpendicular to the planet's face. Directions "left", "right" are determined by the "forward" and "up" directions. The man can neither dig nor fly, so he can move only on the planet's surface in the forward direction, but he is able to turn to the left or to the right thus changing his forward direction (and "left" and "right" too). Since the man's body is a parallelepiped we can speak about the "front", the "rear", the "left", the "right", the "top" and the "bottom" facets, the "rear left", the "rear right", the "front bottom", etc. edges and "top front left", etc. vertices.

To do one step forward his body had to turn 90 degrees around his front bottom edge as depictured in figure 1. His front facet becomes the bottom facet, his top facet becomes the front facet, etc.

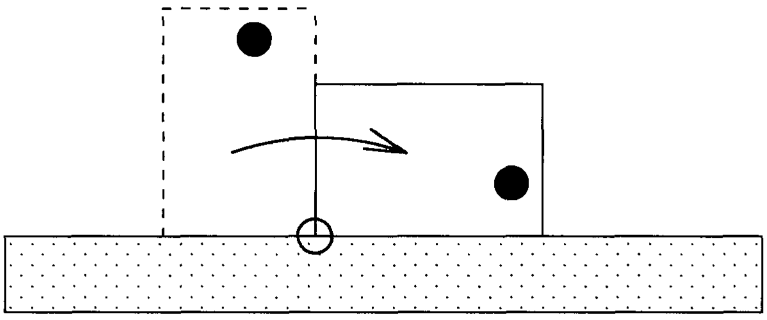

Figure 1: step forward (the right view)

Mikhail can also turn right or left. When he turns he does not make a move. To turn left he turns 90 degrees left around the rear left edge of his body. To turn right, he turns 90 degrees right around the rear right edge of his body. His right turn is shown in figure 2.

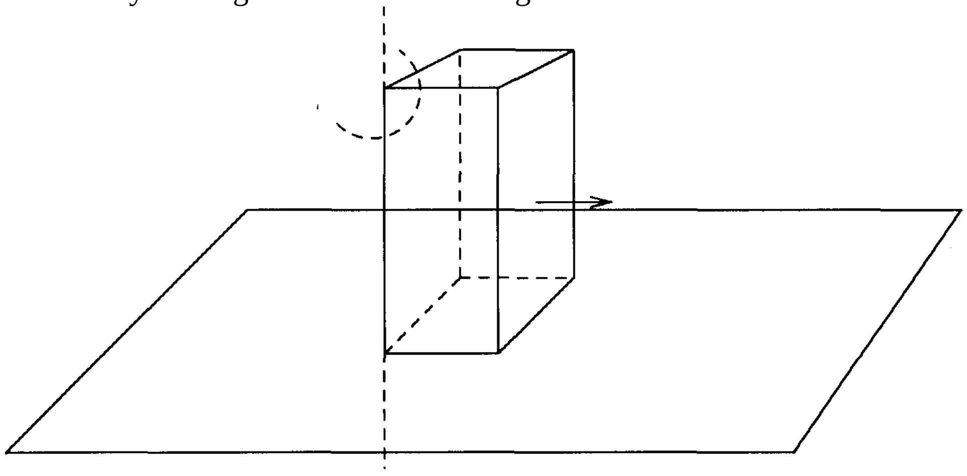

Figure 2: Turn right

Sometimes during his walk he steps onto a planet's edge. He can walk on the edge, can turn left or right provided that his body has one or more common points with the planet. Sometimes he has to change a planet's face (thus changing the "up" direction). This could happen only if he has to make a step forward and after that step his front bottom edge hangs out (i. e. has no common points with) the planet's surface. The planet's facet change is not counted as a step forward and he still has to turn around his front bottom edge to complete the move. He changes the planet's face by turning around the planet's edge under his bottom face as shown in figure 3.

Mr. Kubachev can change more than one planet's face during one step, but he never turns right or left if his turn edge hangs out of the planet's surface.

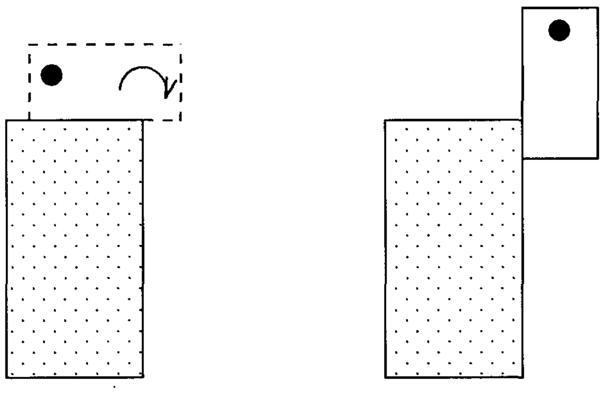

Figure 3: Step around the world's edge (right view)

His spot (marked point) retains the same relative position on Mikhail's body but moves somehow in space and changes its position relatively to the planet. The task is to determine the position of Kubachev's spot after the given number of movements and turns.

2 Input

The input for the program is given in the text file named A.IN, located in the current directory. Your program should be able to accept the input in free format, i.e. with arbitrary number of spaces, newlines and tabulations between items. The file starts with a decimal number designating the number of walks, hereafter in referred as data sets, to process.

Input data for each walk is as follows: three integer numbers to be the coordinates *x*, *y*, *z* of the first corner point of the planet's parallelepiped. The next three integer numbers are the coordinates *x*, *y*, *z* of the opposite point of the planet. The next six integers are the coordinates $x_1, y_1, z_1, x_2, y_2, z_2$ two opposite points of the man's body. The edges of both parallelepipeds are parallel to the coordinate axis. The next three integer numbers are the coordinates of the spot on the man's body and the last three integer numbers are the movement direction vector. The movement direction vector has the length of 1 and is parallel to one of the coordinate axis. You can assume, that the coordinates and the direction vector are correct and that you can determine the "up" direction at the start point of the man's walk. The movement commands follows the figures' description. They defined as follows:

F	move one step forward
L	turn left, don't move
R	turn right, don't move
S	stop the movement

Each movement command consists of a single character as explained in the table. Command S is always the last command in the data set and terminates the data set.

3 Output

Your program should use the text file A.OUT to output the results. The unique line of the output file contains the coordinates of the spot.

4 Example

4.1 Sample input

```
0    0    0
10   10   10
5    5    10
7    7    12
6    6    12
1    0    0
F
     L F F R

       F F S
```

4.2 Sample output

```
11   5    8
```

Problem B: The Cubical Universe

1 Description

The fantastic cubical universe consists of a number of worlds. Each world has the form of parallelepiped and they can intersect each other. Your program has to calculate the volume of the intersection of the worlds in the universe.

2 Input

The input data are located in the text file B.IN. The input has free format as well. Data starts with an integer number *k* that indicates the number of the worlds in that universe. Each world is specified with six non-negative numbers x^0, y^0, z^0, x^1, y^1, z^1 that designate the coordinates of the opposite vertices of the world parallelepiped. The world's edges are parallel to the coordinate axes. You may suppose that the volume of the intersection $v < 2^{31}$.

3 Output

Your program should use the text file B.OUT to output its results. It should be consist of the unique phrase:

- “This universe is not complete.” if the worlds have no common points;
- “This universe is perfect.” if the volume of the intersection of the worlds is 666;
- “The intersection volume is *v* cubical light years.” otherwise.

4 Example

4.1 Sample input:

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

4.2 Sample output:

```
The intersection volume is 1 cubical light years.
```

Problem C: Text Formatting

1 Description

You need to develop the algorithm to format a text so it has sharp left and right margins. You can use only monospace fonts, i.e. the fonts in which every character (including space) has the equal width, so the position of a character on the line is measured in width units, hereafter in referred as columns which are numbered from zero.

2 Input

The first line of program's input (located in text file C.IN) consists three numbers *s*, *r*, *i*:

- ▪ *s*: $0 \le s$ is the left margin of the text;
- ▪ *r*: $s < r \le 78$ is the right margin of the text;
- ▪ *i*: $-s \le i < r - s - 1$ is the paragraph indentation level.

The text to format locates in the rest lines of program's input. It consists of paragraphs, which are separated with one or more empty lines.

A paragraph consists of words and punctuation marks. A word is a sequence of alphanumeric symbols A-Z, a-z, 0-9 (small and capital Latin letters and digits) terminated with a space or a punctuation mark. A punctuation mark is one of the following: ".", ",", “:”, “;”, “!”, “?”, “(“, “)”, “-“. Spaces symbols are spaces, tabulation and newline characters (i.e., chr(13)+chr(10)) if they do not form an empty line. All space characters in a paragraph are ignored and do not affect the formatting.

3 Output

The output should be located in the text file C.OUT and should start with the line containing the following information

"The text contains *p* paragraphs, *w* words, *m* punctuation marks."

where *p* is the number of paragraphs, *w* is the number of words, and *m* is the number of punctuation marks in the text. Second line should be empty. The formatted text should start from the third line and meet the following regulations:

1. 1. The first line of the paragraph should start in column $s + i$ and should end in column $r - 1$. The next lines, except the last, should start in column *s* and end in column $r - 1$. The last line should start in column *s* but may end in any column $c \le r - 1$;
2. 2. One word should be separated from another word by one or more spaces or a line break;
3. 3. Punctuation marks ".", ",", ":", ";", "!", "?", ")" are not separated from the previous word but are separated from the next word;
4. 4. Punctuation mark "(" is separated from the previous word but is not separated from the next word;
5. 5. Punctuation mark "-" is separated from the both sides;
6. 6. No text line can start with a punctuation mark, except "(";
7. 7. Spaces between the words in the line should be uniformly distributed, i.e. the number of spaces should be either *k* or *k* +1 where *k* is the value, calculated by your program and different for different lines of text;
8. 8. The number of spaces between two words in the same line do not increase from left to right in the line;
9. 9. The paragraph should occupy the minimal number of lines;

10. 10. The paragraphs should be separated with one empty line;
11. 11. The text should not contain trailing spaces after the last paragraph.

You may assume, that the format that meets the regulations given above exists. The formatted text will always contain at least two words in all lines except the last lines of the corresponding paragraphs. A paragraph never starts with a punctuation mark (except “(“).

4 Example

4.1 Sample input:

```
10 40 3
Now if you
                are reading the
    output of the program that might mean
that    your program works   (at least in somewhat).


     But do          not hurry to send this program to judges! Check
yourself one more time!



Your program will be rejected anyway. .    . You are still hoping? What
are you doing here?
Go home and do not
load your head with this stupid stuff            !
```

4.2 Sample output:

```
-----
The text contains 3 paragraphs, 63 words, 11 punctuation marks.

             Now if you are reading the
          output of the program that might
          mean that your program works (at
          least in somewhat).

             But do not hurry to send this
          program to judges! Check yourself
          one more time!

             Your program will be rejected
          anyway... You are still hoping?
          What are you doing here? Go home
          and do not load your head with
          this stupid stuff!
```

Problem D: The Cubical Races

1 Description

The second favorite amusement for the cubical citizens after the cubical casino is the cubical races. The races are held on the specially built racing tracks. A racing car is not a very powerful machine with speed limitations, and, unfortunately, it is not equipped with brakes. The only way to stop is to slow down speed gradually. During the practice day (Friday before the races' Sunday) car pilots practicing on the track and trying to find the optimal route for this track. The famous pilot Mikhael Cubaher asked you to write a program to help him to find the optimal path on the track.

To model the real races we will use a discrete approximation. The race field is the rectangle with the size $w * h$ ($l \le w \le 128$, $l \le h \le 128$) (coordinates x, y are integer numbers, the point at location x, y can belong either to the track or to the dirt around the track). The car starts its race from the point (0,0) and runs to the given point. The car makes one move per second. Before the move it can increase or decrease its v_x or v_y speed component by one unit, and then move on the track accordingly to its new speed. Both v_x and v_y cannot be changed on the same turn simultaneously. The new position after the move is $x^* = x + v_x$, $y^* = y + v_y$ provided the point x^*, y^* who must belong to the track. The car has to reach the destination point in the minimal number of turns. The car's speed at the moment of reaching the destination point does not matter. The maximal speed of the car is that $|v_x| \le 3$ & $|v_y| \le 3$.

2 Input

The input data are specified in the text file named D.IN. The first line consists of four integer numbers w, h, x_d, y_d where w is the width of the race field, h is the height of the race field and x_d, y_d $(0 \le x_d < w, 0 \le y_d < h)$ are the coordinates of the target point.

Each of the next h lines of the input data describes the one row of race field. It contains w characters. Character '.' (dot) means that this point of the field belongs to the track and character '*' (asterisk) means that the point is the dirt around the track. The start point (0,0) is located in the left bottom corner of the field and always belongs to the track.

3 Output

Your program should use the text file D.OUT to output its results. Your program's output should contain the unique phrase "The destination point has reached in *n* turns." where *n* is the minimal number of turns needed to reach the destination point from the start point. If the destination point is not accessible, the phrase "The destination point cannot be reached." should be printed.

4 Example

4.1 Sample input:

```
6    6    5    5
*****.
*.....
*.****
*.****
*.****
..****
```

4.2 Sample output:

```
The destination point has reached in 9 turns.
```

Problem E: The Bug in the Pyramid

1 Description

The top-secret laboratory called "Laboratory for Supernatural Defense" (LSD) decided to conduct a revolutionary research. This research is devoted to the supernatural properties of pyramids (tetrahedrons). Previously they did a similar research and obtained very interesting results that indicated that a razor put into a pyramid for some time was mysteriously sharpened.

The goal of this research should be to analyze the impact of pyramids on alive objects like cockroaches, mice and (later) humans to confirm the idea that intellectual level of an animal rises drastically when the animal is placed into a pyramid.

The scheme of the experiment as follows: a cockroach is put on the surface of a pyramid. Then a special message is transmitted into the pyramid using carefully chosen wave length. This message transmits the coordinates of a point on the surface of this pyramid in so-called pyramidal encoding. The animal is expected to run on the surface to the designated point using the shortest path.

To control the experiment and to evaluate its results they need the software that could find the shortest path between two points on the pyramid. All calculations will be proceeded with the exactitude of $1*10^{-5}$.

2 Input

The input starts is in the text file E.IN and consists of 15 real numbers. First 9 numbers are the coordinates of three vertices of the pyramid (the forth vertex is always (0,0,0)) given as three triples (x, y, z). The next three numbers are the coordinates of the bug on the pyramid. The last three numbers are the coordinates of the destination point the bug should run to. All coordinates are given in orthogonal Cartesian coordinate system.

3 Output

Output should be placed in the text file named E.OUT and should be contained one line.

If the coordinates of the bug are inside the pyramid your program should print "The bug position is inside the pyramid."

If the coordinates of the bug are outside the pyramid, "The bug position is outside the pyramid." should be printed.

If the coordinates of the destination point lie inside the pyramid, "The target position is inside the pyramid." to be printed.

Finally, "The target position is outside the pyramid." has to be printed if the destination point lies outside the pyramid.

If none of these exceptional situations happens your program should print the shortest distance between two points on the pyramid with four significant digits after the decimal point as shown in the example.

4 Example

4.1 Sample input:

```
1    0    0
0    1    0
0    0    1
0.5  0.5  0
1    0    0
```

4.2 Sample output:

```
The minimal distance is 0.7071
```