

Problem Set for the 3rd Day of Summer Camp

Japanese Alumni Group

Contest Held on: 24 Sep 2006

This problem set contains seven problems identified by A through G.

Problems B through G were newly created by the members of Japanese Alumni Group. You may use this problem in any form, entirely or in part, with or without modification, for any purposes, without prior or posterior consent to Japanese Alumni Group, provided that your use is made solely at your own risk.

Problem A was taken from ACM ICPC Northeast North America 2005. This problem was modified by the members of Japanese Alumni Group for clarification of the problem statement, adjustment of the difficulty level in solving the problems, attempt to hide the source of the problems from the contestants during the contest proper, and/or other purposes. The source of this problem was revealed to the contestants at the review session held after the contest proper. *The permission to use stated above is not applicable to these problems.*

THE PROBLEM SET IS PROVIDED "AS-IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN NO EVENT SHALL JAPANESE ALUMNI GROUP, THE MEMBERS OF THE GROUP, OR THE CONTRIBUTORS TO THE GROUP BE LIABLE FOR ANY DAMAGE ARISING IN ANY WAY OUT OF THE USE OF THE PROBLEM SET, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Problem A

Rolling Cubes

Input: A.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

A wooden block is standing on a large board with one-inch squares. The block is one inch deep, two inches wide and four inches tall. First, the block is sitting on exactly two squares. The top of the block is black, the bottom is white, the front is green, the back is red, the left face is blue, and the right face is yellow.

Then the block is moved to a new location on the board by being rolled. The block is rolled in such a way that one edge is always in contact with a line of the board at all times. The block never slides.

There are four basic operations that can be performed on the block: F (rolling the block forward), B (backwards), L (to the left), and R (to the right).

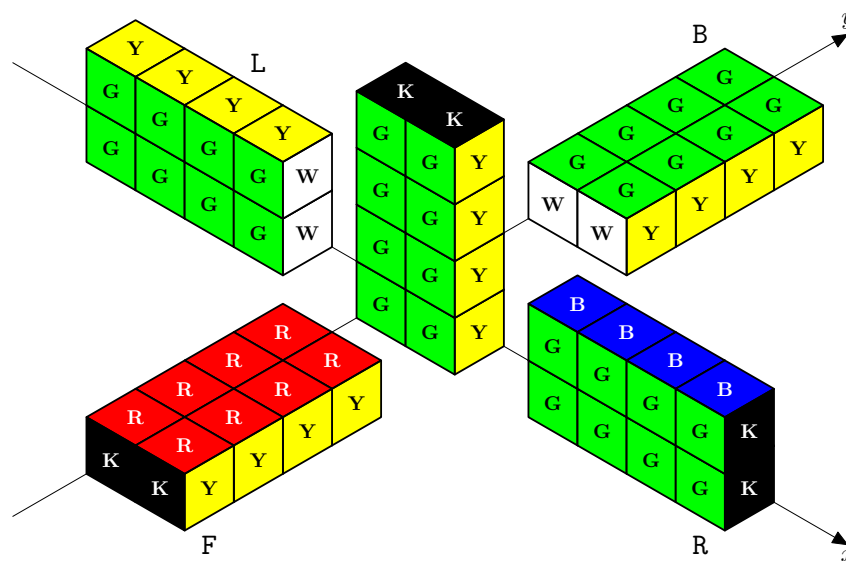


Figure 1: The rolling cubes

The diagram shows the original position of the block (center) and the aspect but not the exact board location of the block after each of the four basic operations.

Your task is to write a program that outputs the resulting state of the block, given a sequence of basic operations.

Input

The input contains a series of data sets. Each data set consists of a single line that contains a string of up to 1000 letters from the set 'F', 'B', 'L', and 'R'.

Output

For each data set, your program should follow the given sequence of basic operations, then print the new location of the front left corner of the block relative to the original location and the colors on the front, left, and top of the block. The output by your program should imitate the sample output below.

Sample Input

FRBL

Output for the Sample Input

x: -2, y: 0, front: yellow, left: black, top: red

Problem B

Reading Brackets in English

Input: B.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

Shun and his professor are studying Lisp and S-expressions. Shun is in Tokyo in order to make a presentation of their research. His professor cannot go with him because of another work today.

He was making final checks for his slides an hour ago. Then, unfortunately, he found some serious mistakes! He called his professor immediately since he did not have enough data in his hand to fix the mistakes.

Their discussion is still going on now. The discussion looks proceeding with difficulty. Most of their data are written in S-expressions, so they have to exchange S-expressions via telephone.

Your task is to write a program that outputs S-expressions represented by a given English phrase.

Input

The first line of the input contains a single positive integer N , which represents the number of test cases. Then N test cases follow.

Each case consists of a line. The line contains an English phrase that represents an S-expression. The length of the phrase is up to 1000 characters.

The rules to translate an S-expression are as follows.

1. An S-expression which has one element is translated into “a list of *<the element of the S-expression>*”. (e.g. “(A)” will be “a list of A”)
2. An S-expression which has two elements is translated into “a list of *<the first element of the S-expression>* and *<the second element of the S-expression>*”. (e.g. “(A B)” will be “a list of A and B”)
3. An S-expression which has more than three elements is translated into “a list of *<the first element of the S-expression>*, *<the second element of the S-expression>*, ... and *<the last element of the S-expression>*”. (e.g. “(A B C D)” will be “a list of A, B, C and D”)
4. The above rules are applied recursively. (e.g. “(A (P Q) B (X Y Z) C)” will be “a list of A, a list of P and Q, B, a list of X, Y and Z and C”)

Each atomic element of an S-expression is a string made of less than 10 capital letters. All words (“a”, “list”, “of” and “and”) and atomic elements of S-expressions are separated by a single space character, but no space character is inserted before comma (“,”). No phrases for empty S-expressions occur in the input. You can assume that all test cases can be translated into S-expressions, but the possible expressions may not be unique.

Output

For each test case, output the corresponding S-expression in a separate line. If the given phrase involves two or more different S-expressions, output “AMBIGUOUS” (without quotes).

A single space character should be inserted between the elements of the S-expression, while no space character should be inserted after open brackets (“(”) and before closing brackets (“)”).

Sample Input

4
a list of A, B, C and D
a list of A, a list of P and Q, B, a list of X, Y and Z and C
a list of A
a list of a list of A and B

Output for the Sample Input

(A B C D)
(A (P Q) B (X Y Z) C)
(A)
AMBIGUOUS

Problem C Grated Radish

Input: C.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

Grated radish (daikon-oroshi) is one of the essential spices in Japanese cuisine. As the name shows, it's made by grating white radish.

You are developing an automated robot for grating radish. You have finally finished developing mechanical modules that grates radish according to given instructions from the microcomputer. So you need to develop the software in the microcomputer that controls the mechanical modules. As the first step, you have decided to write a program that simulates the given instructions and predicts the resulting shape of the radish.

Input

The input consists of a number of test cases. The first line on each case contains two floating numbers R and L (in centimeters), representing the radius and the length of the cylinder-shaped radish, respectively. The white radish is placed in the xyz -coordinate system in such a way that cylinder's axis of rotational symmetry lies on the z axis.

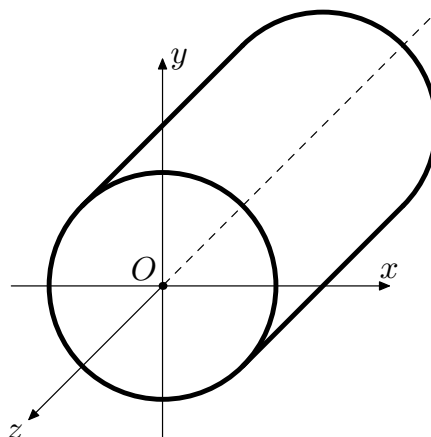


Figure 2: The placement of the white radish

The next line contains a single integer N , the number of instructions. The following N lines specify instructions given to the grating robot. Each instruction consists of two floating numbers θ and V , where θ is the angle of grating plane in degrees, and V (in cubic centimeters) is the volume of the grated part of the radish.

You may assume the following conditions:

- the direction is measured from positive x axis (0 degree) to positive y axis (90 degrees),
- $1 \leq R \leq 5$ (in centimeters),
- $1 \leq L \leq 40$ (in centimeters),
- $0 \leq \theta < 360$, and
- the sum of V 's is the smaller than the volume of the given white radish.

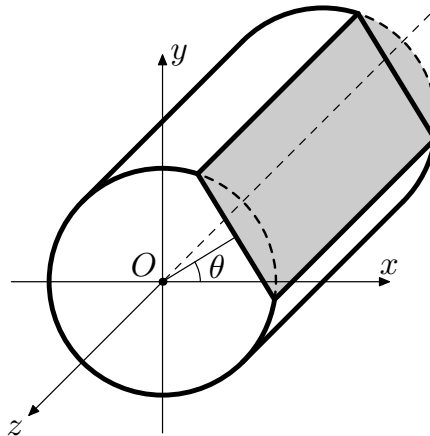


Figure 3: An example of grating

Output

For each test case, print out in one line two numbers that indicate the shape of the base side (the side parallel to xy -plane) of the remaining radish after the entire grating procedure is finished, where the first number of the total length is the linear (straight) part and the second is the total length of the curved part.

You may output an arbitrary number of digits after the decimal points, provided that difference from the true answer is smaller than 10^{-6} centimeters.

Sample Input

```

2
1 2
1
42 3.141592653589793
5 20
3
0 307.09242465218927
180 307.09242465218927
90 728.30573874452591

```

Output for the Sample Input

```

2.0 3.141592653589793
8.660254038 5.235987756

```

Problem D Greedy, Greedy.

Input: D.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

Once upon a time, there lived a dumb king. He always messes things up based on his whimsical ideas. This time, he decided to renew the kingdom's coin system. Currently the kingdom has three types of coins of values 1, 5, and 25. He is thinking of replacing these with another set of coins.

Yesterday, he suggested a coin set of values 7, 77, and 777. "They look so fortunate, don't they?" said he. But obviously you can't pay, for example, 10, using this coin set. Thus his suggestion was rejected.

Today, he suggested a coin set of values 1, 8, 27, and 64. "They are all cubic numbers. How beautiful!" But another problem arises: using this coin set, you have to be quite careful in order to make changes efficiently. Suppose you are to make changes for a value of 40. If you use a greedy algorithm, i.e. continuously take the coin with the largest value until you reach an end, you will end up with seven coins: one coin of value 27, one coin of value 8, and five coins of value 1. However, you can make changes for 40 using five coins of value 8, which is fewer. This kind of inefficiency is undesirable, and thus his suggestion was rejected again.

Tomorrow, he would probably make another suggestion. It's quite a waste of time dealing with him, so let's write a program that automatically examines whether the above two conditions are satisfied.

Input

The input consists of multiple test cases. Each test case is given in a line with the format

$$n \ c_1 \ c_2 \ \cdots \ c_n$$

where n is the number of kinds of coins in the suggestion of the king, and each c_i is the coin value.

You may assume $1 \leq n \leq 50$ and $0 < c_1 < c_2 < \cdots < c_n < 1000$.

The input is terminated by a single zero.

Output

For each test case, print the answer in a line. The answer should begin with "Case # i : ", where i is the test case number starting from 1, followed by

- "Cannot pay some amount" if some (positive integer) amount of money cannot be paid using the given coin set,
- "Cannot use greedy algorithm" if any amount can be paid, though not necessarily with the least possible number of coins using the greedy algorithm,
- "OK" otherwise.

Sample Input

```
3 1 5 25
3 7 77 777
4 1 8 27 64
0
```

Output for the Sample Input

Case #1: OK

Case #2: Cannot pay some amount

Case #3: Cannot use greedy algorithm

Problem E First Experience

Input: E.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

After spending long long time, you had gotten tired of computer programming. Instead you were interested in electronic construction. As your first work, you built a simple calculator. It can handle positive integers of up to four decimal digits, and perform addition, subtraction and multiplication of numbers. You didn't implement division just because it was so complicated. Although the calculator almost worked well, you noticed that it displays unexpected results in several cases. So, you decided to try its simulation by a computer program in order to figure out the cause of unexpected behaviors.

The specification of the calculator you built is as follows. There are three registers on the calculator. The register *R1* stores the value of the operation result of the latest operation; the register *R2* stores the new input value; and the register *R3* stores the input operator. At the beginning, both *R1* and *R2* hold zero, and *R3* holds a null operator. The calculator has keys for decimal digits from '0' to '9' and operators '+', '-', '×' and '='. The four operators indicate addition, subtraction, multiplication and conclusion, respectively. When a digit key is pressed, *R2* is multiplied by ten, and then added by the pressed digit (as a number). When '+', '-' or '×' is pressed, the calculator first applies the binary operator held in *R3* to the values in *R1* and *R2* and updates *R1* with the result of the operation. Suppose *R1* has 10, *R2* has 3, and *R3* has '-' (subtraction operator), *R1* is updated with 7 (= 10 - 3). If *R3* holds a null operator, the result will be equal to the value of *R2*. After *R1* is updated, *R2* is cleared to zero and *R3* is set to the operator which the user entered. '=' indicates termination of a computation. So, when '=' is pressed, the calculator applies the operator held in *R3* in the same manner as the other operators, and displays the final result to the user. After the final result is displayed, *R3* is reinitialized with a null operator.

The calculator cannot handle numbers with five or more decimal digits. Because of that, if the intermediate computation produces a value less than 0 (i.e., a negative number) or greater than 9999, the calculator displays "E" that stands for *error*, and ignores the rest of the user input until '=' is pressed.

Your task is to write a program to simulate the simple calculator.

Input

The input consists of multiple test cases.

Each line of the input specifies one test case, indicating the order of key strokes which a user entered. The input consists only decimal digits (from '0' to '9') and valid operators '+', '-', '*', and '=' where '*' stands for '×'.

You may assume that '=' occurs only at the end of each line, and no case contains more than 80 key strokes.

The end of input is indicated by EOF.

Output

For each test case, output one line which contains the result of simulation.

Sample Input

1+2+3+4+5+6+7+8+9+10=
1000-500-250-125=
1*2*3*4*5*6=
5000*5000=
10-100=
100*100=
10000=

Output for the Sample Input

55
125
720
E
E
E
E

Problem F Web 0.5

Input: F.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

In a forest, there lived a spider named Tim. Tim was so smart that he created a huge, well-structured web. Surprisingly, his web forms a set of equilateral and concentric N -sided polygons whose edge lengths form an arithmetic sequence.

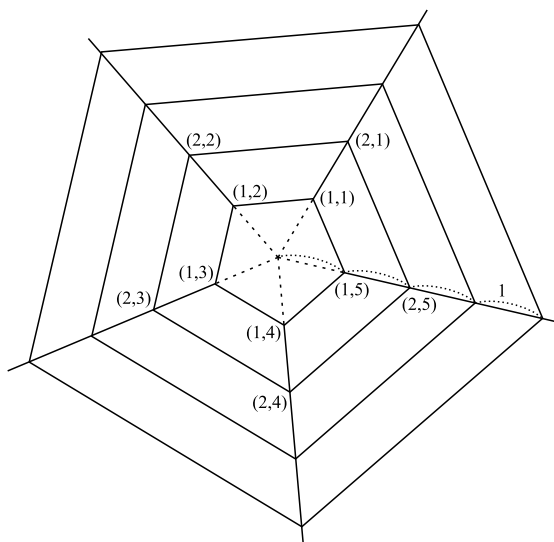


Figure 4: An example web of Tim

Corresponding vertices of the N -sided polygons lie on a straight line, each apart from neighboring vertices by distance 1. Neighboring vertices are connected by a thread (shown in the figure as solid lines), on which Tim can walk (you may regard Tim as a point). Radial axes are numbered from 1 to N in a rotational order, so you can denote by (r, i) the vertex on axis i and apart from the center of the web by distance r . You may consider the web as infinitely large, i.e. consisting of infinite number of N -sided polygons, since the web is so gigantic.

Tim's web is thus marvelous, but currently has one problem: due to rainstorms, some of the threads are damaged and Tim can't walk on those parts.

Now, a bug has been caught on the web. Thanks to the web having so organized a structure, Tim accurately figures out where the bug is, and heads out for that position. You, as a biologist, are interested in whether Tim is smart enough to move through the shortest possible route. To judge that, however, you need to know the length of the shortest route yourself, and thereby decided to write a program which calculates that length.

Input

The input consists of multiple test cases. Each test case begins with a line containing two integers N ($3 \leq N \leq 100$) and X ($0 \leq X \leq 100$), where N is as described above and X denotes the number of damaged threads. Next line contains four integers r_S, i_S, r_T, i_T , meaning that Tim is starting at (r_S, i_S) and the bug is at (r_T, i_T) . This line is followed by X lines, each containing four integers r_A, i_A, r_B, i_B , meaning that a thread connecting (r_A, i_A) and (r_B, i_B) is damaged. Here (r_A, i_A) and (r_B, i_B) will be always neighboring to each other. Also, for all vertices (r, i) given above, you may assume $1 \leq r \leq 10^7$ and $1 \leq i \leq N$.

There will be at most 200 test cases. You may also assume that Tim is always able to reach where the bug is.

The input is terminated by a line containing two zeros.

Output

For each test case, print the length of the shortest route in a line. You may print any number of digits after the decimal point, but the error must not be greater than 0.01.

Sample Input

```
5 1
2 1 3 2
2 1 2 2
0 0
```

Output for the Sample Input

```
4.18
```

Problem G Philosopher's Stone

Input: G.txt

Summer Camp (3rd Day), Tokyo
24 Sep 2006

In search for wisdom and wealth, Alchemy, the art of transmutation, has long been pursued by people. The crucial point of Alchemy was considered to find the recipe for *philosopher's stone*, which will be a catalyst to produce gold and silver from common metals, and even medicine giving eternal life.

Alchemists owned Alchemical Reactors, symbols of their profession. Each recipe of Alchemy went like this: gather appropriate materials; melt then mix them in your Reactor for an appropriate period; then you will get the desired material along with some waste.

A long experience presented two natural laws, which are widely accepted among Alchemists. One is the law of conservation of mass, that is, no reaction can change the total mass of things taking part. Because of this law, in any Alchemical reaction, mass of product is not greater than the sum of mass of materials. The other is the law of the directionality in Alchemical reactions. This law implies, if the matter A produces the matter B, then the matter A can never be produced using the matter B.

One night, Demiurge, the Master of Alchemists, revealed the recipe of philosopher's stone to have every Alchemist's dream realized. Since this night, this ultimate recipe, which had been sought for centuries, has been an open secret.

Alice and Betty were also Alchemists who had been seeking the philosopher's stone and who were told the ultimate recipe in that night. In addition, they deeply had wanted to complete the philosopher's stone earlier than any other Alchemists, so they began to solve the way to complete the recipe in the shortest time.

Your task is write a program that computes the smallest number of days needed to complete the philosopher's stone, given the recipe of philosopher's stone and materials in their hands.

Input

The input consists of a number of test cases.

Each test case starts with a line containing a single integer n , then n recipes follow.

Each recipe starts with a line containing a string, the name of the product. Then a line with two integers m and d is given, where m is the number of materials and d is the number of days needed to finish the reaction. Each of the following m lines specify the name and amount of a material needed for reaction.

Those n recipes are followed by a list of materials the two Alchemists initially have. The first line indicates the size of the list. Then pairs of the material name and the amount are given, each pair in one line.

Every name contains printable characters of the ASCII code, i.e. no whitespace or control characters. No two recipe will produce the item of the same name. Each test case always include one recipe for "philosopher's-stone," which is wanted to be produced. Some empty lines may occur in the input for human-readability.

Output

For each case, output a single line containing the shortest day needed to complete producing the philosopher's stone, or "impossible" in case production of philosopher's stone is impossible.

As there are two Alchemists, they can control two reaction in parallel. A reaction can start whenever the materials in recipe and at least one Alchemist is available. Time spent in any acts other than reaction, such as rests, meals, cleaning reactors, or exchanging items, can be regarded as negligibly small.

Sample Input

3
philosopher's-stone
4 10
tongue 1
tear 1
dunkelheit 1
aroma-materia 1

aroma-materia
4 5
solt 1
holy-water 1
golden-rock 1
neutralizer 2

neutralizer
1 1
red-rock 5

7
tongue 1
tear 1
dunkelheit 1
solt 2
holy-water 3
golden-rock 2
red-rock 10

2
philosopher's-stone
5 3
ninjin 1
jagaimo 3
tamanegi 1
barmont 12

barmont
3 1
curry 5
komugiko 20
batar 10

7
ninjin 3
jagaimo 2
tamanegi 5
curry 150
komugiko 750
batar 200

0

Output for the Sample Input

16
impossible
