1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem A

Pattern Matching Using Regular Expression

Input file: regular.in

A regular expression is a string which contains some normal characters and some meta characters. The meta characters include,

`.`	means any character
`[c1-c2]`	means any character between c1 and c2 (c1 and c2 are two characters)
`[^c1-c2]`	means any character not between c1 and c2 (c1 and c2 are two characters)
`*`	means the character before it can occur any times
`+`	means the character before it can occur any times but at least one times
`\`	means any character follow should be treated as normal character

You are to write a program to find the leftmost substring of a given string, so that the substring can match a given regular expression. If there are many substrings of the given string can match the regular expression, and the left positions of these substrings are same, we prefer the longest one.

Input

Every two lines of the input is a pattern-matching problem. The first line is a regular expression, and the second line is the string to be matched. Any line will be no more than 80 character. A line with only an “end” will terminate the input.

Output

For each matching problem, you should give an answer in one line. This line contains the string to be matched, but the leftmost substring that can match the regular expression should be bracketed. If no substring matches the regular expression, print the input string.

Sample Input

```
.*
asdf
f.*d.
sefdfsde
[0-9]+
asd345dsf
[^\*-\*]
**asdf**fasd
b[a-z]*r[s-u]*
abcdefghijklmnopqrstuvwxyz
[T-F]
dfkgjf
end
```

Output for the Sample Input

```
(asdf)
se(fdfsde)
asd(345)dsf
**(a)sdf**fasd
a(bcdefghijklmnopqrstu)vwxyz
dfkgjf
```

1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem B

Maximum Sum

Input file: sum.in

Given a cube of positive and negative integers, find the sub-cube with the largest sum. The sum of a cube is the sum of all the elements in that cube. In this problem, the sub-cube with the largest sum is referred to as the maximal sub-cube.

A sub-cube is any contiguous sub-array of size 1×1 or greater located within the whole array.

As an example, if a cube is formed by following 3×3×3 integers:

```
0 -1 3
-5 7 4
-8 9 1

-1 -3 -1
2 -1 5
0 -1 3

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

Then its maximal sub-cube which has sum 31 is as follows:

```
7 4
9 1

-1 5
-1 3

3 2
-2 1
```

Input

Each input set consists of two parts. The first line of the input set is a single positive integer N between 1 and 20, followed by N ×N×N integers separated by white-spaces (newlines or spaces). These integers make up the array in a plane, row-major order (i.e., all numbers on the first plane, first row, left-to-right, then the first plane, second row, left-to-right, etc.). The numbers in the array will be in the range [-127,127].

The input file is terminated by a value 0 for N.

Output

The output is the sum of the maximal sub-cube.

Sample Input

```
3
0 -1 3
-5 7 4
-8 9 1
-1 -3 -1
2 -1 5
0 -1 3
3 1 -1
1 3 2
1 -2 1
0
```

Output for the Sample Input

```
31
```

1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem C

Minimal Circle

Input file: circle.in

You are to write a program to find a circle which covers a set of points and has the minimal area. There will be no more than 100 points in one problem.

Input

The input contains several problems. The first line of each problem is a line containing only one integer N which indicates the number of points to be covered. The next N lines contain N points. Each point is represented by x and y coordinates separated by a comma. After the last problem, there will be a line contains only a zero.

Output

For each input problem, you should give a one-line answer which contains three numbers separated by commas. The first two numbers indicate the x and y coordinates of the result circle, and the third number is the radius of the circle.

Sample Input

```
2
0.0,0.0
3,0
5
0,0
0,1
1,0
1,1
2,2
0
```

Output for the Sample Input

```
1.5,0,1.5
1,1,1.414
```

1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem D

Elevator Simulation

Input file: eleva.in

You are to simulate the movements of an elevator, which works according to the following rules:

1. The elevator stays 3 seconds for customers to get in or get out. If some customers want to get in and some want to get out, the elevator stays 6 seconds.
2. It takes the elevator 2 seconds to go one story upwards or downwards.
3. When there is no request for service, the elevator stays where it is. Namely, it is idle.
4. When the elevator is idle, and a request comes from upwards or downwards, it moves upwards or downwards. If a request comes from the story where it stays, the elevator opens its door to serve these customers.
5. When the elevator goes upwards or downwards, it keeps its direction until there is no request from upwards or downwards. However, it breaks off to let customers get out or pick up customers if necessary.
6. If the elevator is to go upwards or downwards, it does not accept customers intending to go upwards or downwards.
7. When the elevator can act more than one way according to aforementioned rules, it chooses the action with the highest priority. The priorities for actions, from the highest to the lowest, are:
   let customers get out
   let customers get in
   go downstairs
   go upstairs

Initially (at time 0), the elevator is in the 0th story, and it is idle. Given the requests, you are to output how the elevator moves in a designated interval. Suppose that each request comes at an integer second.

Input

The input file begins in its first line with the number of stories of a building, and other two numbers representing an interval. These three numbers are separated by a blank.

Each of the following lines in the input file consists of three numbers, N1, N2, and N3, representing a request coming from N2th story to N3th story at time N1.

A line of three 0's separates two test cases. After the last test case, there is a line containing three 0's.

Output

You have to output what the elevator acts at each second in the designated interval.

Sample Input

```
5 9 67
10 2 1
13 2 4
21 4 2
26 4 3
40 1 2
45 0 3
46 0 2
0 0 0
0 0 0
```

Output for the Sample Input

```
9: idle at story 0
10: Going up to 1
11: Going up to 1
12: Going up to 2
13: Going up to 2
14: Let upstair-customers get in at story 2
15: Let upstair-customers get in at story 2
16: Let upstair-customers get in at story 2
17: Going up to 3
18: Going up to 3
19: Going up to 4
20: Going up to 4
21: Let customers get out at story 4
22: Let customers get out at story 4
23: Let customers get out at story 4
24: Let downstair-customers get in at story 4
25: Let downstair-customers get in at story 4
26: Let downstair-customers get in at story 4
27: Going down to 3
28: Going down to 3
29: Let customers get out at story 3
30: Let customers get out at story 3
31: Let customers get out at story 3
32: Going down to 2
33: Going down to 2
34: Let customers get out at story 2
35: Let customers get out at story 2
36: Let customers get out at story 2
37: Let downstair-customers get in at story 2
38: Let downstair-customers get in at story 2
39: Let downstair-customers get in at story 2
```

```
40: Going down to 1
41: Going down to 1
42: Let customers get out at story 1
43: Let customers get out at story 1
44: Let customers get out at story 1
45: Going down to 0
46: Going down to 0
47: Let upstair-customers get in at story 0
48: Let upstair-customers get in at story 0
49: Let upstair-customers get in at story 0
50: Going up to 1
51: Going up to 1
52: Let upstair-customers get in at story 1
53: Let upstair-customers get in at story 1
54: Let upstair-customers get in at story 1
55: Going up to 2
56: Going up to 2
57: Let customers get out at story 2
58: Let customers get out at story 2
59: Let customers get out at story 2
60: Going up to 3
61: Going up to 3
62: Let customers get out at story 3
63: Let customers get out at story 3
64: Let customers get out at story 3
65: idle at story 3
66: idle at story 3
67: idle at story 3
```

1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem E

Card Game — Kitty Fishing

Input file: card.in

There is a simple two-player card game called “Kitty Fishing”. When the game begins, player A and B have the same number of cards. Then each gives out one card in turn. Each card given out on the table should be laid overlapped one by one. When the card newly given out finds a card which has the same value on the table, the player who gives out the card will take the cards between the two same cards following the order the cards on the table, and put them to the back of his cards. The same player continues to give out next cards. Player giving a card called a turn. Note: Do not change the order of your cards.

The following is an example.

At the beginning:
A has cards 1, 4, 2, 3 and B has cards 2, 1, 3, 4

First: A gives out 1.
A: 4, 2, 3
B: 2, 1, 3, 4
Cards on the table: 1.

Second: B gives out 2.
A: 4, 2, 3
B: 1, 3, 4
Cards on the table: 1, 2

Third: A gives out 4.
A: 2, 3
B: 1, 3, 4
Cards on the table 1, 2, 4

Forth: B gives out 1.
A: 2, 3
B: 3, 4
Cards on the table: 1, 2, 4, 1

The card ‘1’ given out by B is the same as the first one of the cards on the table. So B takes the cards following the turn of ‘1, 4, 2, 1’. Then it will be:
A: 2, 3
B: 3, 4, 1, 4, 2, 1
Cards on the table: NULL

In this example, A and B have four turns.

If one of the players has given out all of his cards, he will lose the game, and the other one is the winner. The game is over.

Write a program that will play the game of “Kitty Fishing”.

Input

The input file contains one or more data sets. Each data set consists of three lines: The first line contains an integer which gives out the turns you should play, and the next two lines are cards which A and B have. Each one’s cards will be ended with the number 0.

A line which contains a single 0 will end the input file. No input lines follow that line.

Output

If the game has been over before the turns, just write out the winner’s name. If the game has not been over, write out the cards two players have in the order in their hands and the cards on the table.

Sample Input

```
30
5 8 5 6 5 7 3 4 6 2 1 7 7 1 2 1 0
3 8 7 8 8 5 6 3 6 2 3 2 4 4 1 4 0
0
```

Output for the Sample Input

```
Case 1:
1 5 7 3 5 3 3 6 6 1 1
4 8 8 8 5 8 6 7 5 6 2 2 4 7 2 1 3 4
2 7 4
```

1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem F

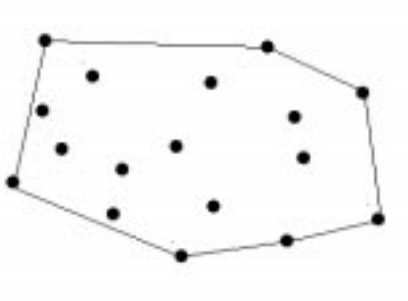
Surround the Trees

Input file: tree.in

There are a lot of trees in an area. A peasant wants to buy a rope to surround all these trees. So at first he must know the minimal required length of the rope. However, he does not know how to calculate it. Can you help him?

The diameter and length of the trees are omitted, which means a tree can be seen as a point. The thickness of the rope is also omitted which means a rope can be seen as a line.

There are no more than 100 trees.



Input

The input file contains one or more data sets. Each input data set consists of a series of coordinates of the trees. Each coordinate is a positive integer pair, and each integer is less than 32767. Each pair is separated by ‘,’. The series is ended with 0,0.

The data set which starts with 0,0 terminates the input for your program.

Output

The minimal length of the rope. The precision should be 10^{-2}.

Sample Input

```
12,7 24,9
30,5 41,9 80,7
50,87 22,9 45,1 50,7 0,0
0,0
```

Output for the Sample Input

```
243.06
```

1997-1998 Asia Regional
ACM International Collegiate Programming Contest

Problem G

Employment Planning

Input file: employ.in

A project manager wants to determine the number of the workers needed in every month. He does know the minimal number of the workers needed in each month. When he hires or fires a worker, there will be some extra cost. Once a worker is hired, he will get the salary even if he is not working. The manager knows the costs of hiring a worker, firing a worker, and the salary of a worker. Then the manager will confront such a problem: how many workers he will hire or fire each month in order to keep the lowest total cost of the project.

Input

The input file may contain several data sets. Each data set contains three lines. First line contains the months of the project planed to use which is no more than 12. The second line contains the cost of hiring a worker, the amount of the salary, the cost of firing a worker. The third line contains several numbers, which represent the minimal number of the workers needed each month. The input file is terminated by line containing a single ‘0’.

Output

The output contains one line. The minimal total cost of the project.

Sample Input

```
3
4 5 6
10 9 11
0
```

Output for the Sample Input

```
199
```