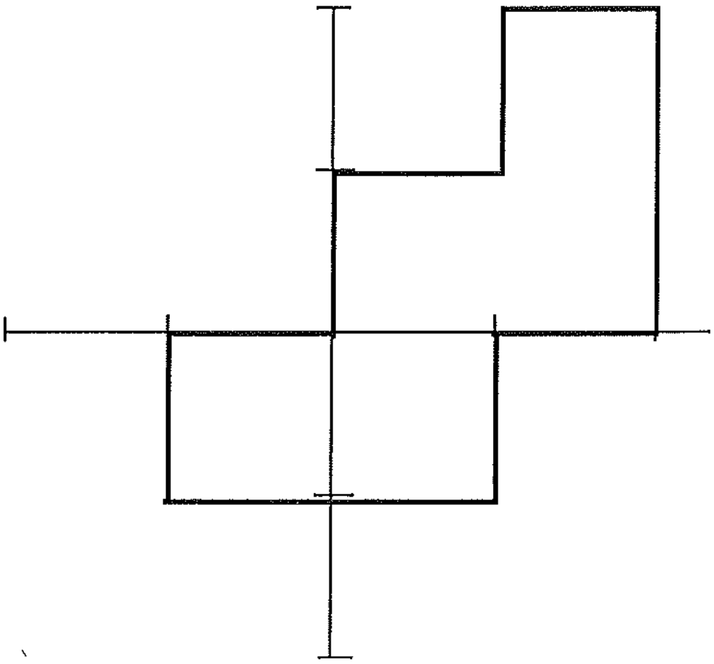
Problem T **Tracing Objects** 150 points

As part of a robot vision project you have been asked to write a program to construct an outline of an object from a series of given corner points. All the corners are right angles, but they could be convex or concave angles. All the sides are parallel to the x- or y-axis. There are no interior holes in the objects. A typical object could look like:



All the corner points have integer coordinates, but unfortunately, due to the way this particular system works, they are in random order. Your program must locate the top right corner (the point which, among the points with the greatest x coordinate, has the greatest y coordinate) and then trace the outline in a clockwise direction.

Input will be from a file called PROBLEMT.DAT and will consist of several objects to trace. For each object, the first line will give the number of points in the contour, N, and will always be an integer less than 100. After this will follow N lines with two integers on each line, separated by one or more spaces. These give the coordinates of the corner points. No coordinate will be greater than 10,000. The file will be terminated by a line containing an N value of 0.

Output, which must be written to standard output (the screen), must be, for each object given in the input, firstly a line containing the number N of points for that object, then the N points given in the input (in the same format as the input) but ordered so that they give the outline of the figure traced in a clockwise direction. You may assume the points given will always define a valid object.

EXAMPLE (the figure shown above)

```
Input                     Output
10                        10
1 0                       2 2
1 2                       2 0
-1 -1                     1 0
2 2                       1 -1
0 1                       -1 -1
0 0                       -1 0
-1 0                      0 0
2 0                       0 1
1 -1                      1 1
1 1                       1 2
0
```

Problem U Shangri-LaLaaLLaL 150 points

In the remote South Pacific, there is a group of islands with some unusual traditions about names. These islands are very beautiful, and an anthropologist friend of yours has offered you the chance of accompanying her on fieldwork there if you help her to analyse the names used on these islands. The thought of some months in a tropical paradise with your friend, however, is a minor consideration next to the interest of this problem.

On each of these islands, every man must have an exact number of sons, no more and no less; two on one island, three on another; the highest number on any island is 4. On each island this tradition has been observed without fail since it started; no boy or man has ever died before having the correct number of sons. On each island the entire male population traces itself back to an original father figure (now dead). Each son of this father figure took a name which is transliterated by a single letter - for example, on one island, the names were F, Q, A and Z (the names were always different). Each son of these sons was given a name which was their father's name, with one of the adopted names appended - for example, Q's sons were QF, QQ, QA and QZ. Every male since then has had the same number of sons and named them by appending one of the same set of names to his own name (so QA's sons were QAF, QAQ, QAA and QAZ). Thus the men on the island have names like FZQQQZZF.

Actually the men are called "Tom", "Dick", "Harry" and so on; the "real" names are kept secret. However, at a ceremony every year, the names of all of the living patriarchs are chanted in alphabetic order, with no breaks to show where each name begins and ends. (Note that a patriarch is the oldest living male in a family; when one dies, all of his sons become patriarchs.) Your anthropologist friend wants you to decipher the chanted compound names to find what the real names of the living patriarchs are.

Input is from a file PROBLEMU.DAT and will consist of a series of joined patriarchs' names, each from a different island. Each joined name could be up to 400 characters, with up to four different upper-case letters used; the joined name will be terminated by an exclamation mark(!). Each name could take several lines; no line will be longer than 60 characters in length. The end of the file is the end of the input.

Your program must output two lines for each joined name. The first line should be the alphabetically least shortest patriarch's name. The second line should be the alphabetically least longest patriarch's name. You may assume there is a solution for each joined name. Note that "alphabetic order" is the same for the transliterated names (using English letters) as it is for the actual names.

EXAMPLE

Input

```
AFQZ!
AAABAABBBABBABB
BAABBBABBBBB!
AAABAABBABCACBABBBCC!
```

Output

```
A
A
AA
BBBAA
C
ABA
```

Problem V Space Missions 150 points

NASA has started planning for unmanned interstellar missions and has asked you to help with this preliminary planning. They are planning to land two automatic probes on each planet and have a surface vehicle travel from one to the other. They want to know how much fuel to provide for the surface travel, given any parameters for the distance the vehicle has to travel, and slowness with which it can travel (the atmosphere could be as thick as honey and the terrain full of deep vertical holes). The vehicle will be able to carry only a limited amount of fuel; if it cannot carry enough to travel between the probes in one trip, NASA are planning to operate the mission as follows:

- The vehicle will leave the probe loaded with fuel, travel a distance and leave some of its fuel at a fuel depot, retaining just enough to return. It could do this several times.

- When it has enough fuel at the first fuel depot, it will create a second fuel depot, using the first depot to refuel (if necessary) on the outward trips, or the inward trips, or both.

- This process of leaving fuel depots continues, until it is able to get a fuel depot far enough away, and have enough fuel in each depot along the way, so that it can reach the other probe in a single trip, refuelling at each fuel depot.

The fuel is packaged in containers and only whole containers can be left in fuel depots. The amount of fuel used to travel to each fuel depot must also be a whole number of containers, as loading or unloading with a partially used fuel container on board could be dangerous (the vehicle uses batteries when it is not moving).

For any given distance between the probes (measured in containers of fuel used for the travel) and any given capacity of the vehicle, NASA want to know what is the least total amount of fuel which can be used to make the journey. For example, if travelling between the probes takes 25 fuel containers, and the vehicle can carry 15 containers, then the journey can be made with as few as 59 containers (and no fewer) by creating three fuel depots according to the following table:

Distance from start	2	5	10
Amount of fuel when first created	10	9	5

Input is from a file PROBLEMV.DAT and will be lines describing possible scenarios, each containing two integers; the first gives the distance which must be travelled (measured in fuel containers), and the second gives the number of containers the vehicle can carry. Each number will be less than 1000 and greater than 2. The end of the file is the end of the input.

For each input line you must output one line, either giving the least number of containers needed for the scenario, or (if the scenario takes more than 1,000,000 containers) the words "Too many".

EXAMPLE

Input

```
150 200
25 15
17 3
375 210
10 4
```

Output

```
150
59
Too many
1047
172
```

Problem W **Find the number** 150 points

Quite complex sets can be formed by taking intervals of numbers and forming the union and intersection of them. The object of this problem is to find if a certain number is in one of these intersections and unions.

Input will be from the file PROBLEMW.DAT and will be a series of lines. Each line will start with an integer (between -2,000,000,000 and 2,000,000,000), followed by a blank, followed by a set expression. Each set expression defines a set, using standard mathematical intervals (a,b) = {x; a < x < b}, (a,b] = {x; a < x <= b}, [a,b) = {x; a <= x < b}, [a,b] = {x; a <= x <= b} (a and b will always be integers between -2,000,000,000 and 2,000,000,000), and union (u), intersection (n) and set difference (-) signs. Brackets will be used as in normal expressions; the expressions will be fully bracketed so you do not need to worry about precedence of operators. There may be any amount of imbedded spaces (not inside numbers, of course), and the only non-space characters which will occur will be digits, -, n, u, (,), [,]. For example, (((1, 4) n [2, 6)) u [7, 10)) - (3, 8) represents the set of numbers x such that $2 \leq x \leq 3$ or $8 \leq x < 10$. You may assume the expression is always valid, but there could be any number of superfluous (and) characters around subexpressions. The input will be terminated by a line containing a single 0.

Output must be one line for each line of input, giving the word "Yes" or "No" depending on whether the integer is in the set or not.

EXAMPLE

Input

```
1234567890 (-1234567890 , 1234567890) u (1234567890  ,2000000000)
6 [2,6]n[6,20]
4 (((1,4)n[2,6))u[7,10)) - (3,8)
0
```

Output

```
No
Yes
No
```