The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest

for World Finals

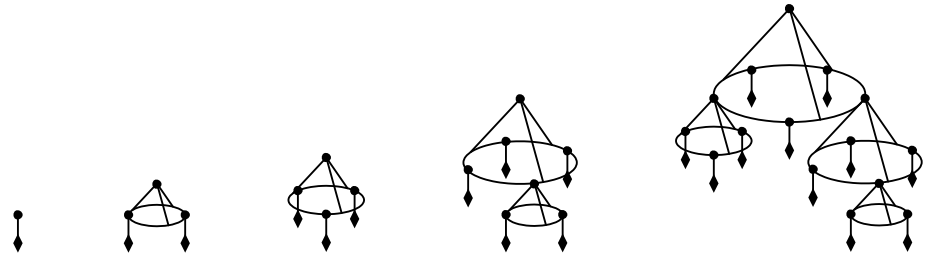
Problem A

Chandelier

Input File: chandelier.in

Lamps-O-Matic company assembles very large chandeliers. A chandelier consists of multiple levels. On the first level crystal pendants are attached to the rings. Assembled rings and new pendants are attached to the rings of the next level, and so on. At the end there is a single large ring — the complete chandelier with multiple smaller rings and pendants hanging from it.

A special-purpose robot assembles chandeliers. It has a supply of crystal pendants and empty rings, and a stack to store elements of a chandelier during assembly. Initially the stack is empty. Robot executes a list of commands to assemble a chandelier.



On command "`a`" robot takes a new crystal pendant and places it on the top of the stack. On command "`1`" to "`9`" robot takes the corresponding number of items from the top of the stack and consecutively attaches them to the new ring. The newly assembled ring is then placed on the top of the stack. At the end of the program there is a single item on the stack — the complete chandelier.

Unfortunately, for some programs it turns out that the stack during their execution needs to store too many items at some moments. Your task is to optimize the given program, so that the overall design of the respective chandelier remains the same, but the maximal number of items on the stack during the execution is minimal. A pendant or any complex multi-level assembled ring count as a single item of the stack.

The design of a chandelier is considered to be the same if each ring contains the same items in the same order. Since rings are circular it does not matter what item is on the top of the stack when the robot receives a command to assemble a new ring, but the relative order of the items on the stack is important. For example, if the robot receives command "`4`" when items $\langle i_1, i_2, i_3, i_4 \rangle$ are on the top of the stack in this order (i_1 being the topmost), then the same ring is also assembled if these items are arranged on the stack in the following ways: $\langle i_2, i_3, i_4, i_1 \rangle$, or $\langle i_3, i_4, i_1, i_2 \rangle$, or $\langle i_4, i_1, i_2, i_3 \rangle$.

Input

The first line of the input file contains a single integer N that indicates the number of programs. Each of the next N lines contains a valid program for the robot. The program consists of at most 10,000 characters.

Output

For each program, you should write two lines. On the first line write the minimal required stack capacity (number of items it can hold) to assemble the chandelier. On the second line write some program for the assembly robot that uses stack of this capacity and results in the same chandelier.

Write a blank line between two instances.

Sample Input	Output for the Sample Input
1 aaaaa3aaa2aaa45	6 aaa3aaa2aaa4aa5

The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest for World Finals

Problem B
Dancing Cube
Input File: cube.in

A *dancing cube* is built with eight hollow smaller cubes called unit blocks. Each inner face of the unit blocks is colored white, blue, yellow or red. Every unit block has exactly one white face and one red face opposite to each other. The dancing cube is placed on a horizontal floor, so that its four vertical faces look toward north, south, east and west.

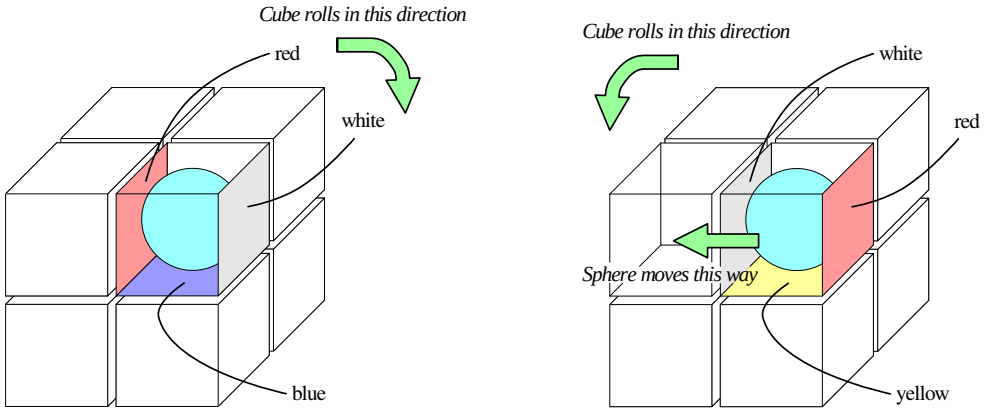
We put a *magical sphere*, whose size fits a unit block, into one of the unit blocks. Then, according to the bottom color of the unit block which the magical sphere is in, the dancing cube and the magical sphere act as described below. After the action, depending on a new state of the magical sphere and the dancing cube, they act over again. We call a series of the actions a *dance* of the cube. The dance stops when the magical sphere goes out of the dancing cube.

When the bottom is red, the magical sphere jumps up to the upper unit block. If the sphere is already in one of the upper unit blocks, it jumps out of the dancing cube causing stop of the dance.

When the bottom is white, the magical sphere falls into the lower unit block. If the sphere is already in the lower, it bounces on the floor and springs up to the upper block.

When the bottom face is colored blue, the dancing cube rolls in the direction of the vertical white face from the magical sphere. In this case, the sphere does not move to another block.

When the bottom face is colored yellow, the magical sphere first moves toward the vertical white face, then the dancing cube rolls in the direction of the vertical white face from the sphere before its movement. In case the sphere moves out of the dancing cube, the cube makes the last roll before it stops the dance.



Your task is to write a program that simulates the dance of the cube and prints the direction which the dancing cube rolls in.

Input

The input consists of a set of test cases each of which contain nine lines.

The first line of each case contains three letters, separated by a space, which indicates the initial position of the magical sphere. The first letter is either `T` or `B`, the second either `N` or `S`, the third either `E` or `W`. Here `T`, `B`, `N`, `S`, `E` and `W` represent upside, downside, north, south, east and west respectively.

The next eight lines describe blocks in the following order: the upper-northeastern, upper-northwestern,

upper-southeastern, upper-southwestern, lower-northeastern, lower-northwestern, lower-southeastern and lower-southwestern ones. Each line consists of six letters indicating the color of the top, northern, eastern, western, southern and bottom sides respectively. Each letter is `R` (representing red), `W` (white), `B` (blue) or `Y` (yellow).

The end of input is indicated by end of file.

Output

For each case, print in a line an integer *n*, which represents the number of rolls the dancing cube makes, followed by *n* letters, each of which represents the direction the cube rolls in. Each letter should be one of `N` (representing north), `S` (south), `E` (east) and `W` (west). If the dance of the cube is not stopped, just print `-1` instead.

The samples below may help you with your understanding.

Sample Input	Output for the Sample Input
B S W B Y R W B B W B B Y Y R R B Y Y B W B Y R W B B W B B Y Y R B B W R Y Y R B Y Y B W B W B Y R Y T S E B Y R W B B W B B Y Y R R B Y Y B W B Y R W B B W B B Y Y R B B W R Y Y R B Y Y B W B W B Y R Y	5 N E N N S -1

The 2005 29th Annual **acm** International Collegiate

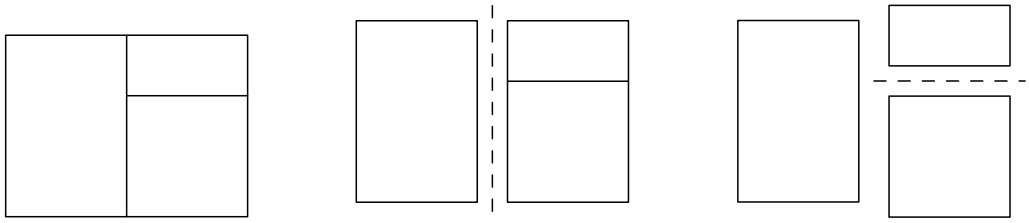
Programming Contest Practice Contest for World Finals

Problem C

Cutting Edge

Input File: cut.in

Uncle Jeff owns a glass shop, which sells glass panes for windows and picture frames. As you probably know, a glass pane can only be cut if the cut goes from edge to edge of the pane in a straight line. The figure below shows how a glass pane can be cut into three smaller glass panes.



(a) a whole glass pane (b) first cut (c) second cut

Uncle Jeff normally operates as follows. He first collects various orders for small rectangular glass panes, for windows or picture frames. He then marks the position of each small rectangular pane onto a big rectangular glass pane, such that no two marked rectangles overlap. Finally, he performs a sequence of horizontal and vertical cuts, always from edge to edge of the pane being cut, so as to produce glass panes for all the customers.

Since the last phase (the actual cutting of the big glass pane into pieces) is the most boring thing one could ever imagine, uncle Jeff is asking you for help. He wants a program which given a big rectangular glass pane and lower-left and upper-right coordinates of each marked rectangle determines the order in which the edge-to-edge cuts can be performed. This list of cuts will be fed into a machine which will do the boring cuts for him!

Input

The input contains several test cases. The first line of a test case contains an integer N indicating the number of windows and picture frames in the test ($2 \leq N \leq 2000$). Each of the next N lines contains four integers X_1, Y_1, X_2, Y_2, where (X_1, Y_1) and (X_2, Y_2) represent the lower-left and upper-right coordinates marked by uncle Jeff on the big glass pane ($-5000 \leq X_1, Y_1, X_2, Y_2 \leq 5000$; $X_1 < X_2$ and $Y_1 < Y_2$). You should assume the following of each test case:

- The marked rectangles do not overlap (but may intersect on the border points) and divide the big glass pane completely into rectangular regions, so that no glass is wasted. This means that the lower-left and upper-right coordinates of the big glass pane can be inferred from the coordinates of the marked rectangles.
- It is possible to split up the big glass pane into the small marked rectangles through a sequence of edge-to-edge cuts.

The end of input is indicated by $N = 0$.

Output

For each test case in the input your program must produce an ordered list of cuts that must be performed to separate the big glass pane into the desired smaller panes. Each cut must appear in a different line. A cut is described by four integers X_1, Y_1, X_2, Y_2, where (X_1, Y_1) and (X_2, Y_2) specify the endpoints of the cut, with $X_1 < X_2$ and $Y_1 = Y_2$ for a horizontal cut and $X_1 = X_2$ and $Y_1 < Y_2$ for a vertical cut. As more than one ordering of cuts may be possible, your program must print the list in a particular order. If at some point more than one cut is possible, print first the cut with smaller X_1; if more than one cut is still possible, print first the one with smaller Y_1. Print a blank line after each test case list.

Sample Input	Output for the Sample Input
3 0 0 20 30 20 0 40 20 20 20 40 30 6 1 2 2 4 2 3 3 5 1 4 2 5 2 2 3 3 3 2 4 3 3 3 4 5 0	20 0 20 30 20 20 40 20 2 2 2 5 1 4 2 4 2 3 4 3 3 2 3 3 3 3 3 5

The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest for World Finals

Problem D

Game

Input File: game.in

There is a legend that mathematicians in the XVIII century enjoyed playing the following game. The game was played by three mathematicians. One of them was the game master. First of all, the game master declared some positive integer number N. After that he chose two different integer numbers X and Y ranging from 1 to N and told their sum to one player and their product to the other player. Each player knew whether he was told the sum or the product of the chosen numbers.

After that the players in turn informed the game master whether they knew the numbers he had chosen. First the player who was told the sum said whether he knew the numbers, after that the player who was told the product did, and so on.

For example the dialog could look like this.

> Game master: "Let N be 10."

After that he chooses two numbers ranging from 1 to 10 and tells their sum to player S and their product to player P.

> Player S: "I don't know these numbers."
> Player P: "I don't know these numbers."
> Player S: "I don't know these numbers."
> Player P: "I don't know these numbers."
> Player S: "Oh, now I know these numbers. You have chosen 3 and 6."

Given N and M — the number of times the players have said "I don't know these numbers," you have to find all possible pairs of numbers that could have been chosen by the game master.

Input

Each line of the input file contains N and M ($2 \le N \le 200$, $0 \le M \le 100$). The input file ends at a line containing two zeros.

Output

For each instance, first output the number of possible pairs of numbers that could have been chosen by the game master from the range 1 to N if both players altogether had said "I don't know these numbers" M times. After that output these pairs one on a line. The first number of each pair must be smaller than the second. The pairs must be sorted in ascending order by the smaller numbers of pairs, then the greater ones. Output a blank line between two instances.

Sample Input	Output for the Sample Input
10 4 0 0	3 2 5 3 6 3 10

The 2005 ACM Programming Contest Practice Contest for World Finals

The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest

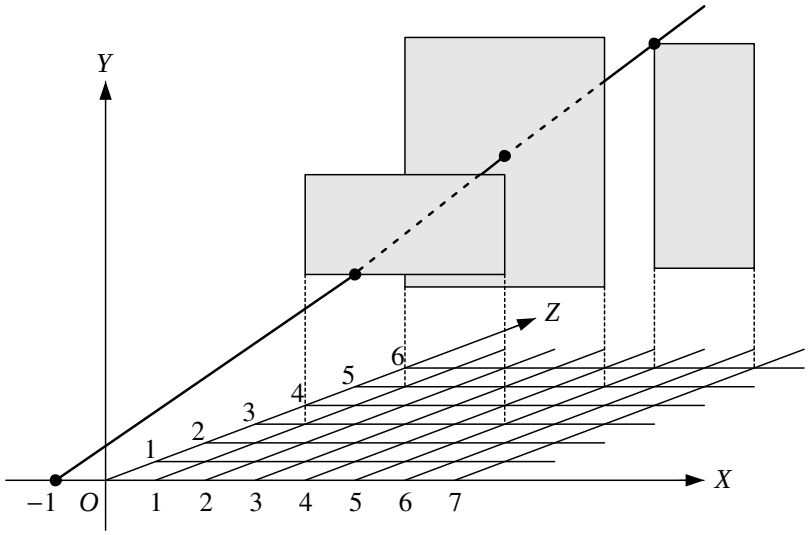
for World Finals

Problem E

Gunman

Input File: gunman.in

Consider a 3D scene with OXYZ coordinate system. Axis OX points to the right, axis OY points up, and axis OZ points away from you. There is a number of rectangular windows on the scene. The plane of each window is parallel to OXY, its sides are parallel to OX and OY . All windows are situated at different depths on the scene (different coordinates $z > 0$).



A gunman with a rifle moves along OX axis ($y = 0$ and $z = 0$). He can shoot a bullet in a straight line. His goal is to shoot a single bullet through all the windows. Just touching a window edge is enough.

Your task is to determine how to make such shot.

Input

The input file consists of a series of test cases.

The first line of each case contains a single integer number n ($2 \le n \le 100$) — the number of windows on the scene. The following n lines describe the windows. Each line contains five integer numbers x_{1i}, y_{1i}, x_{2i}, y_{2i}, z_i ($0 < x_{1i}, y_{1i}, x_{2i}, y_{2i}, z_i < 1000$). Here (x_{1i}, y_{1i}, z_i) are coordinates of the bottom left corner of the window, and (x_{2i}, y_{2i}, z_i) are coordinates of the top right corner of the window ($x_{1i} < x_{2i}, y_{1i} < y_{2i}$). Windows are ordered by z coordinate ($z_i > z_{i-1}$ for $2 \le i \le n$).

The end of the input file is indicated by $n = 0$.

Output

Output a single word "`UNSOLVABLE`" if the gunman cannot reach the goal of shooting a bullet through all the windows.

Otherwise, on the first line output a word "`SOLUTION`". On the next line output x coordinate of the point from which the gunman must fire a bullet. On the following n lines output x, y, z coordinates of the points where the bullet goes through the consecutive windows. All coordinates in the output file must be printed with six digits after decimal point.

Output a blank line between two cases.

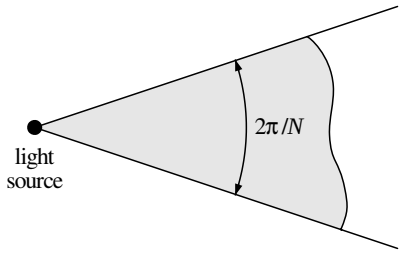
Sample Input	Output for the Sample Input
3 1 3 5 5 3 1 2 5 7 5 5 2 7 6 6 3 2 1 5 4 1 3 5 6 8 2 4 3 8 6 4 0	SOLUTION -1.000000 2.000000 3.000000 3.000000 4.000000 5.000000 5.000000 5.000000 6.000000 6.000000 UNSOLVABLE

The 2005 29th Annual **acm** International Collegiate

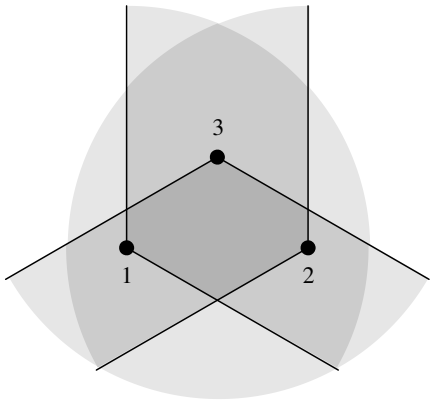
Programming Contest Practice Contest for World Finals

Problem F
Illumination
Input File: ill.in

You are given N light sources on the plane, each of which illuminates the angle of $2\pi/N$ with the vertex in the source point (including its sides).



You must choose the direction of the illuminated angle for each of these sources, so that the whole plane is illuminated. It can be proved that this is always possible.



A light source itself casts no shadow and does not interfere with light beams from the other sources.

Input

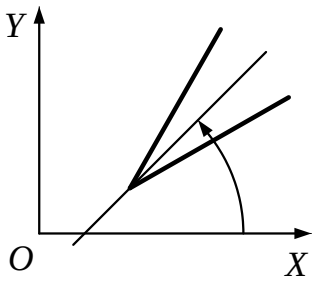
The input file contains multiple test cases.

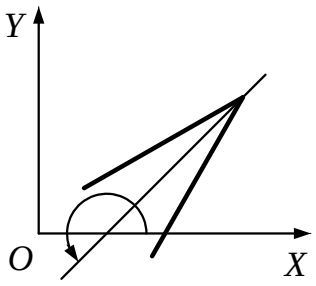
The first line of each case contains N — the number of light sources ($3 \le N \le 30$). Next N lines contain two integer numbers each — the coordinates of the light sources. All coordinates do not exceed 100 by their absolute value. No two sources coincide.

The last line of the input file contains a single zero.

Output

For each test case, print N real numbers — for each light source specify an angle that the bisector of the illuminated angle makes with OX axis, counterclockwise. Print one number per line. Print eight digits after the decimal point. No angle must exceed 100π by its absolute value. Print a blank line between two cases.





Sample Input	Output for the Sample Input
3 0 0 2 0 1 1 0	0.52359878 2.61799388 4.71238898

The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest for World Finals

Problem G

Joke with Turtles

Input File: joke.in

There is a famous joke-riddle for children:

> Three turtles are crawling along a road. One turtle says: “There are two turtles ahead of me.” The other turtle says: “There are two turtles behind me.” The third turtle says: “There are two turtles ahead of me and two turtles behind me.” How could this have happened?
> The answer is — the third turtle is lying!

Now in this problem you have n turtles crawling along a road. Some of them are crawling in a group, so that they do not see members of their group neither ahead nor behind them. Each turtle makes a statement of the form: “There are a_i turtles crawling ahead of me and b_i turtles crawling behind me.” Your task is to find the minimal number of turtles that must be lying.

Let us formalize this task. Turtle i has x_i coordinate. Some turtles may have the same coordinate. Turtle i tells the truth if and only if a_i is the number of turtles such that $x_j > x_i$ and b_i is the number of turtles such that $x_j < x_i$. Otherwise, turtle i is lying.

Input

The input file consists of a set of test cases.

The first line of each case contains integer number n ($1 \le n \le 1000$). It is followed by n lines containing numbers a_i and b_i ($0 \le a_i, b_i \le 1000$) that describe statements of each turtle for i from 1 to n.

$n = 0$ indicates the end of the input file.

Output

For each case, write in a line an integer number m — the minimal number of turtles that must be lying, followed by m integers — turtles that are lying. Turtles can be printed in any order. If there are different sets of m lying turtles, then print any of them. No blank lines should appear.

Sample Input	Output for the Sample Input
3 2 0 0 2 2 2 5 0 2 0 3 2 1 1 2 4 0 0	1 3 2 1 4

The 2005 ACM Programming Contest Practice Contest for World Finals

The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest for World Finals

Problem H

Move the Object

Input File: move.in

Mike and his friends are crazy for a video game. The aim of the game is to move a circular object placed on a two-dimensional field to the destination point by an analog pad. On the field there are some polygonal obstacles through which the object cannot move. When the object gets to the destination point, the player gains the score depending on the distance it moved over, and then advances to the next stage. As you might realize, the shorter move makes the higher score.

Though many obstacles appear in the latter stages, only one obstacle is set in each of first several stages, and Mike is absorbed in moving the object by the shortest distance in those stages. In fact, he moves the object by shorter distance than any his friends, but he is not sure that he really moves by the minimum distance. So he calls you for the help.

Your task is to write a program that computes the minimum moving distance for given data.

Input

The input consists of a series of data sets.

The first line of each set contains an integer N ($3 \leq N \leq 20$) indicating the number of vertices of the polygonal obstacle. The next N lines specify coordinates of the vertices, which are given counterclockwise. Each line consists of two real numbers that indicates x and y-coordinates of a vertex respectively. Then a line containing four numbers SX, SY, DX and DY follows, where (SX, SY) and (DX, DY) are coordinates of the starting and destination points respectively.

Each coordinate value is an integer between -100 and 100 inclusive. The radius of the circular object is one. No obstacle is degenerate. The position of the object is represented by the coordinate of its center. The starting point and the destination point do not coincide.

The input is terminated by $N = 0$.

Output

For each data set, output a line containing the minimum distance needed to move the object from the starting point to the destination point, with four decimal digits and an error not greater than 0.0001.

It is guaranteed that there is a way to move the object from the starting point to the destination point.

Sample Input	Output for the Sample Input
4 -5 -5 5 -5 5 5 -5 5 0 -10 0 10 8 10 20 -10 20 -20 10 -20 -10 -10 -20 10 -20 20 -10 20 10 -21 0 21 0 5 0 10 -5 5 -5 0 0 5 5 0 10 0 16 8 0	25.8546 71.4259 10.0000

The 2005 29th Annual **acm** International Collegiate

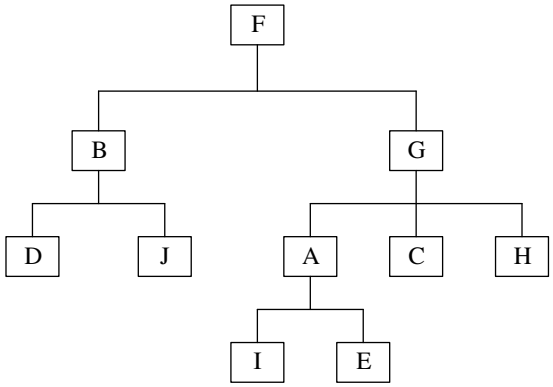
Programming Contest Practice Contest for World Finals

Problem I

Organization Chart

Input File: org.in

An organization chart depicts the structure of the organization. The figure below shows an example of an organization chart.



As you see, an organization chart is considered as a kind of tree. Each edge represents the relationship of a boss and a subordinate, and hence the depth of each node implies the person's position in the organization. Relative position of nodes is also often important; for example, when A and B come just under C, it is important which is placed on the left.

One day, a president passed copies of the organization chart of his company to the two officers X and Y, and ordered them to make *sequences* from the chart in their own ways.

The officer X makes a sequence by tracing edges from right to left, that is, by reverse pre-order traversal. For example, he makes the following sequence from the chart shown above: F, G, H, C, A, E, I, B, J, D.

The officer Y makes another sequence in the order of the levels of the persons. The level of a person is simply measured by his/her depth from the top. As breadth first search, the person of the highest level comes first, while the person of the lowest level comes last. Persons on the equivalent level are ordered from left to right.

The other day, the president has lost the organization chart and all its copies for some reason. But fortunately, he found the sequences made by the officers X and Y, which is enough to reproduce the chart.

Now you are requested to write a program that restructures the organization chart from given two sequences.

Input

Input consists of some data sets. The first line of each set contains a positive integer n ($n \leq 26$) that represents the number of persons in the organization. It is followed by two lines that denote the sequence by the officers X and Y respectively, each containing n capital letters from `A` to `Z` separated by space.

The end of input is indicated by $n = 0$, which should not be processed.

Output

For each set, you should output a line that describes the structure of the organization chart. You should output "`a(bcd)`" if the person `a` is a boss of the persons `b`, `c` and `d` appearing in this order from left to right in the chart. See samples below for the precise format.

Sample Input	Output for the Sample Input
5 A B D C E A B C D E 10 F G H C A E I B J D F B G D J A C H I E 0	A(B(C(E)D)) F(B(DJ)G(A(IE)CH))

The 2005 29th Annual **acm** International Collegiate

Programming Contest Practice Contest for World Finals

Problem J

Poke Poker Pokest

Input File: poker.in

This is a story of a fantasy world named *Upper Earth*. The hero of the story is an explorer Flood, who goes through caves, ruins, deserts, mountains and so on.

One day, he got to the city called *Upper Sea*. At that time the Pork Festival was being held. As soon as he joined the festival, he got to know the name of the best pork dish there — *Pokest*. It tasted so great that no one could eat Pokest without being in heaven. So Flood wanted to eat, but no one knew how to make except that *poking* is the most important factor for that. Fortunately, he soon found that a poker contest would be held for people who want to eat Pokest. Of course he immediately decided to participate the contest, but he was told that only one person could eat; so he had to be a champion of the contest. You may wonder why only one champion could eat. This was because the contest was held by the king of *Upper Sea*, and he could not invite more than one person for security reasons.

Well, there were many contestants in the contest, so the king decided to use a special deck instead of an ordinary fifty-two-card deck. Each card in the deck had a suit and a rank as an ordinary card does, but there were s kinds of suits and r ranks. A suit was represented by a capital letter of first s alphabets (A,B,C,...), and a rank represented by an integer between 1 and r.

A game went as follows. First the game players sat down around the table. After that, the king named a person as the dealer of the game. The person named by the king first chose an arbitary player (including himself or herself) as the first player of the game, then dealt the first five cards in the deck to the first player, the next five cards to the next player (in clockwise order), and so forth. After dealing of cards, each player exchanged any number of cards as he or she wanted, from the first player in clockwise order. Exchange was done by discarding some cards from the hand and drawing the same number of cards from the top of the deck, like an ordinary draw poker. A player could choose to exchange no cards or even all cards in his or her hand. After all player exchanged cards, showdown was carried out, and the player who made the highest hand among the players won the game and was to be invited by the king for a Pokest dish.

In the contest, there were ten kinds of hands shown below in decreasing order. Note that r and 1 were not regarded as consecutive ranks. In addition, no tie breaker is laid down for players of the same hand. If more than one player made the highest hand, no one would be a winner nor could eat Pokest.

Name	Description	Example
Five of a Kind	Five cards of the same rank.	(A,1)(B,1)(C,1)(D,1)(E,1)
Straight Flush	Five cards forming both a straight and a flush.	(A,1)(A,2)(A,3)(A,4)(A,5)
Four of a Kind	Four cards of the same rank.	(A,1)(B,1)(C,1)(D,1)(E,2)
Full House	Three cards of one rank and a pair of another.	(A,1)(B,1)(C,1)(D,2)(E,2)
Flush	Five cards of the same suit.	(A,1)(A,2)(A,5)(A,6)(A,9)
Straight	Five cards of consecutive ranks.	(A,1)(B,2)(C,3)(A,4)(E,5)
Three of a Kind	Three cards of the same rank.	(A,1)(B,1)(C,1)(D,2)(E,5)
Two Pair	Two seperate pairs.	(A,1)(B,1)(C,2)(D,2)(E,5)
One Pair	Two cards of the same rank.	(A,1)(B,1)(C,2)(D,3)(E,5)
No Pair	Cards not forming any hands mentioned above.	(A,1)(B,2)(C,5)(D,7)(A,9)

Now Flood advanced to the final round, and fortunately he was named as a dealer of the final round. He got all cards in the deck and could control the order of the cards by his special ability, but was there a way for him to win...?

Your task is to write a program that determines if there was a winning strategy for a Pokest dish, no matter how many cards the other players exchanged.

Input

Input consists of multiple test cases. The first line of each case contains three integers n ($2 \leq n \leq 100$), s ($1 \leq s \leq 26$) and r ($1 \leq r \leq 100$), where n indicates the number of the players, s indicates the number of suits, and r indicates the number of ranks. The following $10n$ lines contain cards in the deck from the top to the bottom. Input is terminated by end of file.

You can assume that $s \times m \geq 10n$.

Output

For each test case, you should print "`Yes`" in a line if it is possible to eat Pokest, otherwise print "`No`".

Sample Input	Output for the Sample Input
3 5 10 A 1 B 1 C 1 D 1 E 1 A 2 B 2 C 2 D 2 E 10 A 3 B 3 C 3 D 3 E 9 A 4 B 4 C 4 D 4 E 8 A 5 B 5 C 5 D 5 E 7 A 6 B 6 C 6 D 6 A 10	Yes