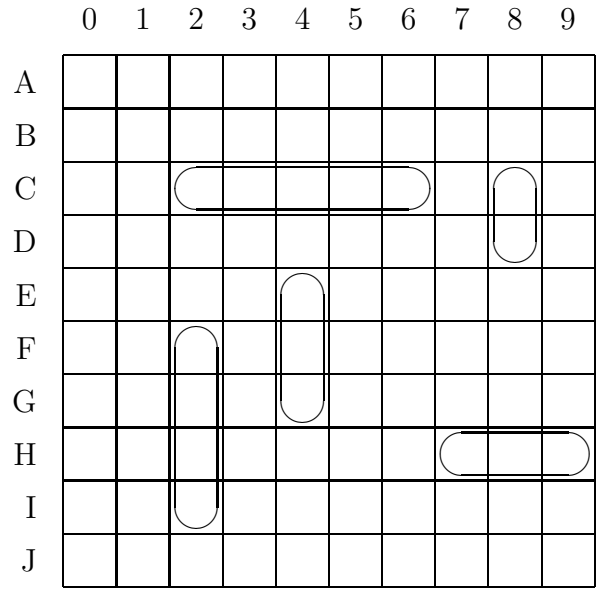
16th Annual Denison Spring Programming Contest
19 February 2005

Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the PC^2 environment.
6. The allowed programming languages are C, C++ and Java.
7. Judges' decisions are to be considered final.
8. There are **six** questions to be completed in **four hours**.

Problem A: All Hands on Derek

Derek loves to play Battleship, and wins almost every time for one very important reason: he cheats. If you will recall, Battleship is played on a 10 by 10 grid, with rows labeled by letters A to J, and columns labeled by integers 0 to 9 (I know, I know its really 1 to 10, but 0 to 9 make things work out easier for us). Each player places 5 ships on the grid. The ship sizes are 5, 4, 3, 3 and 2 grid squares, and they must be placed in either a row or a column (not diagonally). The grid below shows one possible arrangement of the ships:



The object of the game is to sink the other persons ships by "shooting" at various grid locations. After announcing the grid square you are attacking, your opponent tells you whether it was a "hit" (i.e. one of his ships was on that square) or a "miss". If Derek's opponent were to attack grid square C6, an honest player would announce that one of his ships had been hit. But not Derek. He keeps track of all of the shots of his opponent, and constantly picks up and moves his ships to locations which have not hit yet. In this case, he could move the ship one row higher or lower, but not two rows lower since that would bump into another ship (though he could then move that ship) and not one column over to the right, since his boat will still lie on square C6. If his opponent had previously shot at square C1, then Derek could not move his ship one column over to the left either. For this problem you will be given a set of grid squares where shots have been taken, and you need to determine if it's still possible for Derek to place all his ships so that no ship is hit yet.

Input

There will be multiple input sets. Each input set will consist of a single line of the form

$$n\ g_1\ g_2\ g_3\ \ldots\ g_n$$

where n is the number of grid squares which have been shot, and $g_1, \ldots, g_n$ are the square locations. Each location will consists of two characters: an uppercase letter indicating the row

followed by a digit indicating the column. A line which begins with 0 indicates end of input and should not be processed.

Output

For each input set, output either the word "Yes" or "No" depending on whether or not Derek can still place his ships without any being hit or not.

Sample Input

```
1 C4
20 A4 B3 C2 D1 E1 A9 B8 C7 D6 E5 F4 G3 H2 I1 J0 F9 G8 H7 I6 J5
0
```

Sample Output

```
Yes
Yes
```

Problem B: Jiggle Your Squiggle

A squiggle is a string of n letters. Your squiggle is legal if and only if it contains no vowels (a, e, i, o, u). You can make your squiggle legal by giving it a jiggle. Follow these steps to jiggle your squiggle:

- Find the left-most vowel. Suppose it is in position k where positions are numbered from the left starting at 1.
- Rotate the character in position k and every character to the left of position k. To rotate the character in position j, count forward in the alphabet j letters and replace the current character with that new one. If you hit z, then next wrap around to an a and keep going if necessary.
- You must always jiggle from the left to right. So you must go back to the left and fix any new problems (newly appearing vowels), starting with the left-most vowel, before you can proceed to the right.
- Continue to jiggle until your squiggle is legal.

Here is how to jiggle the squiggle `aaa`. First we rotate the left-most `a` (which is in position 1) forward one spot to obtain `baa`. Next we fix the second `a` by rotating it twice and the first character once again: `cca`. Now we rotate the last `a` three spots, the middle `c` two more spots, and the first `c` one more spot: `ded`. But now a vowel has appeared in position 2, which must be jiggled. Continuing on we obtain:

```
aaa > baa > cca > ded > egd > fgd
```

Input

The input consists of integer m followed by m lines containing one squiggle each. Only lower case letters are used in squiggles; no other characters (including whitespaces) are allowed. Any squiggle will contain at most 25 characters ($n \leq 25$). Within this range you are guaranteed to make your squiggle legal.

Output

For each input squiggle, print out the new squiggle after it has been made legal.

Sample Input

```
2
aaa
azaa
```

Sample Output

```
fgd
ghmm
```

Problem C: Elections

Alice, Bob, Carroll, and Dave are founding members of a computer programming club and they are to vote on whether or not to adopt Java as their new official language. However, each member does not have equal voting power; they receive a number of votes based on how many club meetings they have attended. The table shows how many meetings each person has attended:

Alice	24
Bob	17
Carroll	10
Dave	3

There are a total of 54 votes among the four members. Any decision requires more than half the total votes; thus Java will only be adopted if enough people vote together to total at least 28 votes or more; otherwise, the adoption will fail. A coalition is a group of people who all vote the same way. A winning coalition is one in which the total votes of that coalition is more than half (at least 28 in this example). A person is critical to that winning coalition if by removing themselves, the coalition is no longer a winning one.

We can measure how much influence each person has by counting the number of coalitions in which they are critical. In this example, the following are winning coalitions: ABCD, ABC, ABD, ACD, BCD, AB, and AC. The table below shows which members are critical:

Coalition	Alice	Bob	Carroll	Dave
ABCD				
ABC	x			
ABD	x	x		
ACD	x		x	
BCD		x	x	x
AB	x	x		
AC	x		x	
totals	5	3	3	1

Input

The input for this program will consist of a series of input sets, one input set on each line. Each input set will be start with an integer n ($n \leq 10$) representing the number of voters, this is followed by a list of n integers giving the votes for each member. An line with $n = 0$ indicates the end of input.

Output

For each non-empty problem, print a single line of output giving the number of winning coalitions for which each voter is critical.

Sample Input

```
3 3 1 1
4 24 17 10 3
0
```

Sample Output

```
4 0 0
5 3 3 1
```

Problem D: Round and Round We Go

A group of runners always meet evry Saturday at different circuits to run. This group is very unusual in that each runner runs at a different pace. To make sure no one is in trouble, they check in, via a cell phone conference call, every minute. They've noticed that on some circuits they run, at some time they are evenly spaced around the circuit on one of their calls. They think this is pretty cool and so seek out circuit where this happens. You're going to write a program to help them out.

Input

There are multiple input sets. Each input set consists of two lines. Line one contains two integers, m n, where m is the length of the circuit ($m \leq 1000$) and n is the number of runners ($n \leq 20$). A value of $m = 0$ indicates end of input. m will always be a multiple of n. The second line will consist of n integers indicating the per minute pace of each runner. These will all be different.

Output

For each input set produce one line of output Either print the line
`The runners are evenly spaced after` s `minutes.`

where s is the smallest integer where this will happen, or print the line

`No can do.`

if it is impossible to evenly space the runners.

Sample Input

```
4 2
2 4
10 5
1 2 3 4 5
100 10
2 5 4 6 7 13 12 40 1 8
21 3
3 13 17
0 0
```

Sample Output

```
The runners are evenly spaced after 1 minutes.
The runners are evenly spaced after 2 minutes.
No can do.
The runners are evenly spaced after 7 minutes.
```

Problem E: Period

Danny Do Over has created an algorithm that generates random integers. You have serious doubts about Danny's claim. You ask for some data from Danny's algorithm and you want to determine if the data sequence is periodic. Note that a sequence is periodic with period w if for every data item i we have $d_i = d_{w+i}$.

Input

There will be multiple input sets. Each input set is started by an integer n ($n \leq 500$) followed by a line with n integers from Danny's generator. A value of $n = 0$ indicates end of input. You are to search through the data set to determine if there is any possible period in the data set.

Output

For each input set, report one of two responses:

```
No period found.
Possible period of w.
```

where w is the shortest possible period.

Sample Input

Example input:

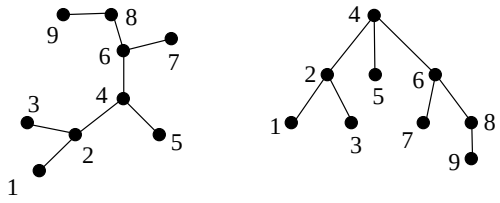
```
11
1 2 3 1 2 3 1 2 3 1 2
7
7 3 6 7 2 3 9
4
1 3 4 1
0
```

Sample Output

```
Possible period of 3.
No period found.
Possible period of 3.
```

Problem F: Hanging a Tree

Mathematically, a *tree* is a connected graph with no cycles. Usually, computer scientists like to draw a tree with one node (called the *root node*) at the top and the rest of the tree hanging below. When given a tree as simply a collection of nodes and edges, we'd like to determine which node to use for the root so the number of edges from the root to the leaf node that is furthest away is as small as possible (so as to not "string out" the display). We say this display has minimum height. The diagram below shows a tree shown as simply a graph and then displayed with a root with minimum height.



Input

There will be multiple input sets. Each set will start with a line containing a single integer n ($n < 21$) which gives the number of nodes in the tree. (A value of $n = 0$ indicates end of input.) Assume these nodes are labeled 1 through n. n lines follow, line j giving the nodes adjacent to node j and is of the form $mv_1v_2 \ldots v_m$ where m is the number of nodes adjacent to node j and $v_1v_2 \ldots v_m$ being those adjacent nodes.

Output

Each input set should produce one line of output of the form
`Make node` k `the root.`
where k is the node that gives a display of minimum height. In case of a tie, use the smallest numbered node.

Sample Input

```
9
1 2
3 1 3 4
1 2
3 2 5 6
1 4
3 4 7 8
1 6
2 6 9
1 8
0
```

Sample Output

```
Make node 4 the root.
```