Rafting

Source file name: rafting.c, rafting.cpp or rafting.java
Input: rafting.in
Output: standard output

A whitewater rafting company is trying to fit its clients into as few rafts as possible. Each raft can take one or two people and has a weight limit. You will be given the weight limit for rafts and the list of clients' weights. Compute the minimum number of rafts needed to accommodate all clients.

Input

The first line of input contains the number of test cases. The test cases follow. Each test case consists of two lines. The first line contains two integers n and w, 1 <= n; w <= 1000. n is the number of clients, w is the weight limit of each raft. The second line contains n integers between 1 and 1000, the weights of the clients.

Output

For each test case output a single line to the standard output. It should contain the minimum number of rafts or the word IMPOSSIBLE if no assignment is possible.

Sample input	**Sample Output**
2	2
3 100	IMPOSSIBLE
95 13 25	
3 100	
1000 1000 1000	

Christmas Trees

Source file name: tree.c, tree.cpp or tree.java
Input: tree.in
Output: standar output

Christmas is always fun at the rebel home base. This year they are buying two Christmas trees and are placing them in elevated stands so that the tops of the trees are level if the trees are of different heights. Given the heights of the trees, you have to print a picture of the two Christmas trees.

Input

The input consists of several lines. Each line will contain two positive decimal integers no larger than 100. These integers will be separated by exactly one space. These integers represent the sizes of the trees; the size of the left tree is followed by the size of the right tree. The last line will contain two zeros, separated by one. This line is not to be processed; it merely signifies the end of the input.

Output

The output cases are to appear in the same order in wich they appear in the input. A full specification is not necessary; the examples will suffice, but here area few pointers: The height of the stem is the same as the input integer; the height of the branches is twice that. The tops of trees are level; the bottoms need not be. There is exactly one space between trees, wich is to say, between the trees is there is exactly one vertical column consisting only of spaces. All trailing spaces are trimmed from each line before printing; the last character of any line in the output file should not be a space. There should be exactly one blanck line following each output case.

Sample input

```
2 3
3 2
0 0
```

Sample Output

```
   *         *
  ***       ***
 *****     *****
*******   *******
   *     *********
   *    ***********
             *
             *
             *

     *         *
    ***       ***
   *****     *****
  *******   *******
 *********     *
***********    *
     *
     *
     *
```

Ball Toss

Source file name: toss.c, toss.cpp or toss.java
Input: toss.in
Output: standar output

Classmates stand in a circle facing inward, each with the direction left or right in mind. One of the students has a ball and begins by tossing it to another student (It doesn't really matter which one).
When one catches the ball and is thinking left, he throws it back across the circle one place to the left (from his perspective) of the person who threw him the ball. Then he switches from thinking left to thinking right. Similarly, if he is thinking right, he throws the ball to the right of the person who threw it to him and then switches from thinking right to thinking left.
There are two exceptions to this rule: If one catches the ball from the classmate to his immediate left and is also thinking left, he passes the ball to the classmate to his immediate right, and then switches to thinking right. Similarly, if he gets the ball from the classmate to his immediate right and is thinking right, he passes the ball to the classmate to his immediate left, and then switches to thinking left (Note that these rules are given to avoid the problem of tossing the ball to oneself).
No matter what the initial pattern of left and right thinking is and who first gets tossed the ball, everyone will get tossed the ball eventually! In this problem, you will figure out how long it takes.
You'll be given the initial directions of n classmates (numbered clockwise), and the classmate to whom classmate 1 initially tosses the ball (Classmate 1 will always have the ball initially).

Input

There will be multiple problem instances. Each problem instance will be of the form n k t1 t2 t3 … tn where n (2 <= n <= 30) is the number of classmates, numbered 1 through n clockwise around the circle, k (> 1) is the classmate to whom classmate 1 initially tosses the ball, and ti (i = 1, 2, … , n) are each either L or R, indicating the initial direction thought by classmate i (n = 0 indicates end of input).

Output

For each problem instance, you should generate one line of output of the form:
Classmate m got the ball last after t tosses.
where m and t are for you to determine. You may assume that t will be no larger than 100,000. Note that classmate number 1 initially has the ball and tosses it to classmate k. Thus, number 1 has not yet been tossed the ball and so does not switch the direction he is thinking.

Sample input	**Sample Output**
4 2 L L L L 4 3 R L L R 10 4 R R L R L L R R L R 0	Classmate 3 got the ball last after 4 tosses. Classmate 2 got the ball last after 4 tosses. Classmate 9 got the ball last after 69 tosses.

Where Is The Ace?

Source file name: ace.c, ace.cpp or ace.java
Input: ace.in
Output: standar output

You come across an apparently respectable man on the street that makes you a seemingly legitimate proposition: the opportunity to play a game of Where Is The Ace. He has a table with three face-up cards in a row: two jokers and one ace, with the ace in the middle. After you make a small wager, he turns the cards face-down and begins to manipulate the cards, swapping cards two at a time. After he completes the swaps, you are then to guess which card is the ace.

The series of card swaps will be given to you in order as a String swaps, containing only the characters L, R, E, and F. The 4 characters indicate the following moves:

L	swap the left and middle cards
R	swap the right and middle cards
E	swap the cards on the ends (the left and right cards)
F	fake swap (no cards actually change position)

You are asked to write a program that calculates the final position of the ace, after all the swaps have been performed.

Input

The first line of the input contains a positive integer N which indicates the number of test cases. The following N lines contain the test cases, one per line. Each test case consists of a single line containing a sequence S of characters L, R, E, and F. The lenght of S is between 0 and 50 inclusive.

Output

For each test case your solution should print, in the order of the test cases, a single line containing: L if the left card is the ace, R if the right card is the ace, and M if the ace is in the middle.

Sample input	Sample Output
5 L REL RFRFR FLRELFLRELLFRLFEELFLRFLELRFLRFREFRFLLRFERLFLREFRFL	L M R M L