Some interesting documents have recently been found in the ACM archives. These documents contain an account of the original version of the ACM ICPC competition, which took place in Bletchley Park, England, in the year 1943.

In this original contest, programs submitted by contestants were executed in a simplified version of the so-called deterministic Turing Machine, which undoubtedly most of you are familiar with. In this simplified version, the tape is not infinitely long. Instead, tape cells are numbered 0 to $10^3 - 1$ (inclusive), from left to right. Also, the alphabet used is the unary alphabet, meaning that, prior to the execution of the program, the tape will be initialized with the string consisting of n ones followed by all zeroes, where n is the numerical value of the input, and after execution the tape contents should be m ones followed by all zeroes, where m is the numerical value of the corresponding output. In the beginning, the tape head is at position 0, and states are also numbered starting with 0, which is assumed to be the initial state. The machines work as usual: for every machine state q and every bit c (0 or 1), there is a at most one rule determining what the next state will be if the symbol at the position of the tape head is c and the current state is q; the rule also specifies the symbol to write at the current position and in which direction the head should move. When no rule applies, the machine stops.

Unfortunately, only the verdicts of the contest judge were found in the archives, there being no trace of the Turing machines submitted or the test cases used. Note that in this original contest, unlike the current one, there was a different verdict for each test case, instead of a general one.

This discovery has caught the attention of the famous adventurer Zaphod Beeblebrox, who intends to launch a project to construct examples of machines and input/output "files" that might have caused these verdicts. Not only have all your efforts to make him understand the futility of such an enterprise failed miserably, but you also have been coerced into writing a program for him that generates such examples. Your program should receive a series of verdicts about one of the Turing machines submitted by the contestants, and return the specifications of a Turing machine, as well as a series of test cases, for which the verdicts are the ones you received.

We consider the output for a machine should be '`MLE`' (memory limit exceeded) if the machine attempts to access any cell outside the tape range specified above; '`TLE`' (time limit exceeded) if it runs for at least 104 iterations without stopping or causing a MLE error; '`WA`' (wrong answer) if the machine stops but returns an incorrect result, and '`AC`' (accepted) if the machine stops and returns the expected result.

Input

The input consists of a certain number (no larger than 30) of series of verdicts. The first line for each contains $1 \leq N \leq 100$, the number of verdicts in the case. The following N lines contain one of the words '`TLE`', '`MLE`', '`WA`' and '`AC`'. The end of input is signaled by a case with $N = 0$, which should not be processed.

Output

For a given test case for your program, your output should adhere to the following format:

The first line contains $1 \leq M \leq 1000$, the number of rules for the machine, and N, the number of test cases for the Turing Machine.

M lines follow, each one containing a rule, in the format $q_{prev}\ c_{prev}\ q_{next}\ c_{next}\ mov$. q_{prev} is the state of the machine before the rule is applied, c_{prev} is the content (either 0 or 1) of the tape at the current position p before the rule is applied, q_{next} is the state of the machine after the rule is applied, c_{next} is the content of position p after the rule is applied, and mov is the direction in which the tape head moves one step after applying the rule ('`L`' if it moves to the left and '`R`' if it moves to the right). The states should be integers between 0 and 1000, inclusive. The Turing machine should be deterministic, that is, there should be at most one transition from any pair (q_{prev}, c_{prev}), and should not be repeated in your output. After this, N lines follow, each one containing two space-separated numbers, X and Y, $1 \leq X, Y \leq 1000$. X is the value of the input to the program, and Y is the value of the expected output.

Note: While this is the recommended syntax, other combinations of new lines and whitespace characters might also get accepted.

Sample Input

```
1
AC
0
```

Sample Output

```
2 1
0 1 0 1 R
0 0 2 1 R
4 5
```