Two little robots are crawling along a string composed of the characters *A* and *B*. Their goal is to determine whether the string is *fine* or *coarse*.

The two robots consider a string to be *fine* if the number of *A*s and the number of *B*s in its middle third are equal. They consider all other strings to be *coarse*. For instance, the string BBABAABBBABA is fine because its middle third, AABB, contains two *A*s and two *B*s; on the contrary, the string BAABABBBBAAB is coarse because its middle third, *ABBB*, contains one *A* and three *B*s. The string AABBAABB is coarse just the same, while it does not have a middle third.

In every moment, each robot occupies one character of the string, and perceives the following things about its surroundings. It can see whether it is standing on the leftmost character, an inner character, or the rightmost character of the string (in other words, it can see whether there are any more characters on the left and right). It can see whether it is alone at its position, or the other robot occupies the same position together with it. Lastly, it can see what character it is standing on, *A* or *B*.

Apart from that, each robot is capable of retaining memories about its travels. Unfortunately, the memory of each robot spans just four bits.

There is a "list of instructions" in every robot's head, and the two lists may differ widely.

Each instruction consists of three parts:

Left side (*condition*)->(*special characters*)Right side (*instruction for action*)

The left side includes:

- One character indicating whether the robot is at one of the string's ends: L for "If it is at the **L**eftmost character", I for "If it is at an **I**nner character", or R for "If it is at the **R**ightmost character"
- One character, O for "if there is **O**ne robot at this position" or T for "if there are **T**wo robots at this position", indicating whether the robot is alone, or the other robot occupies the same position together with it
- The character that the robot is standing on, A for "if it is on **A**", or B for "if it is on **B**"
- 4 characters, each 0 or 1, describing the contents of the robot's memory, e.g., 0101 for "if the memory contains 0101"

In this way, the left side forms a condition, which is a conjunction of these *simple conditions*, and has a value of "true" or "false", depending on the current position of the robot, and the history, stored in its four bits of memory.

The right side appears as follows:

- One character indicating what action the robot is to take: L for "Move one step **L**eft", R for "Move one step **R**ight", S for "**S**tay in place", Y for "Report that **Y**es, the string is fine", or N for "Report that **N**o, the string is coarse"
- Except of the cases when the work is done (actions Y or N) – four more characters, each 0 or 1, indicating the revised contents of the robot's memory, e.g., 0110.

Any character in an instruction line apart from the two special characters and the action character may be replaced with the wildcard ?.

When the wildcard character ? occurs in the left side of the instruction (condition) it means that the corresponding simple condition is not checked.

When the wildcard character ? occurs in the memory overwrite part of an instruction line (i.e. after the action character), it indicates that the corresponding bits of the robot's memory are to be left unchanged.

For instance, the instruction line

LT???01->R??10

tells: "If the robot occupies the leftmost character of the string, the other robot occupies the same position together with it, and the last two bits of its memory are set to 0 and 1, then (no matter whether it is standing on an *A* or a *B*) it is to move one step right and flip the last two bits of its memory to 1 and 0, without changing the first two bits".

Both robots start their journey on the leftmost character of the string. Initially, all bits in each robot's memory are set to 0.

Every millisecond, each robot looks around, searches its memory, consults its list for guidance, and decides to either move one step left, move one step right, stay in place, or voice its opinion about the string.

Additionally, the robot may commit any impressions or intentions to memory. As soon as any robot voices an opinion about the string, their task is done and they switch off. If both robots happen to say their opinion about the string, the opinion of the first is valid.

In detail, each robot checks consecutively the instruction list for an ***exact accordance*** of its current state with the left part of an instruction. When it ***first*** meets one, it executes the instruction found. Otherwise, the robot restarts the search from the beginning of list, and executes the ***first*** instruction containing a wildcard, which corresponds to its position and memory contents. If there is no match again, the robots switch off without recognizing the kind of the string (and their task remains unsolved).

You have to create two instruction lists, one for each robot, such that they will always correctly determine whether a given string of length **at least two characters**, composed of the letters *A* and *B*, is fine or coarse.

This is an **output only** problem. You have to create and submit a single text file robots.txt containing (in this order):

- One line reading Robot 1
- Several instruction lines making up the first robot's instruction list (one instruction in a line)
- One line reading Robot 2
- Several instruction lines making up the second robot's instruction list (one instruction in a line).

The file robots.txt may contain comment lines, which are simply skipped by the robot. Such lines start with the character % (percent).

Constraints

The common number of steps, which the robots can make before announcing the string type, should not exceed 1000 times the string's length.

Each test contains either 16 or 20 strings, each of length at least 12 and at most 3600 characters; each character is A or B.

One test contains only strings of length 12.

Two more tests contain only strings of length at most 36.

Grading

Each test is worth full points if the two robots judge every string in it correctly in a number of steps that does not exceed 1000 times the string's length and 0 points otherwise.

Example

For additional brightening, let us consider a much easier task: let the robots consider "fine" only strings of length 3, with third character A. I. e. only AAA, ABA, BAA and BBA are considered "fine". This can be obtained using a very simple strategy. In fact, one robot is enough for this task, so we can instruct the other to stay in place. "The working one" can move two steps to the right and check if it has reached the rightmost end and if it stands on the letter A.

robots.txt	**Explanation**
Robot 1	For robot 1
???????->S????	Whatever the state – stay in place
Robot 2	For robot 2
L??0000->R0001	If at leftmost end with memory 0000 – go right and make the memory 0001
I??0001->R0010	If on an inner character and with memory 0001 – go right, remembering 0010
R??0001->N	If at the rightmost end with memory 0001 – the string only contains two characters, so say it is "coarse"
R?A0010->Y	If at the rightmost end, on A and with memory 0010 – the string is "fine"
R?B0010->N	If at the rightmost end, on B and with memory 0010 – the string is "coarse"
I??0010->N	If not reached the rightmost end yet, the string is too long, declare it "coarse"

Interpreter

For local testing, you are provided with an interpreter of your file robots.txt, named robots.cpp. Download it in the folder, where you create tour file robots.txt. Using the interpreter, you can test your solution upon strings that you input. It may be run console mode, too, if its compiled version is placed in the same folder, where robots.txt resides. Also, there is a trace mode of the interpreter, when the task is executed step by step, showing the state of the robots and waiting for ENTER from the standard input.

Run without parameters, the interpreter accepts a string from the standard input, upon which to execute the instruction lists, created by you in the robots.txt file, without tracing. If one parameter is given:

- If it is a string of A's и B's, this parameter is checked using the instructions in robots.txt without tracing;
- If it is not recognized as a valid string of A's and B's, the interpreter waits for another string from the standard input and witches the tracing on.

If the parameters are two, the first of them is considered a string to be checked, and the presence of the second switches on the tracing mode.

For example (in consoled mode):

Running robots ABAB	checks the string ABAB with no tracing;
Running robots ABAB 1	checks the string ABAB with tracing;
Running robots 1	waits for console input, which is to be checked with tracing.

Of course, you can use and change the interpreter as you see fit.