Code Jam 2011 - World Finals

Program within a Program

Problem

You have a robot on an infinite east-west highway, and it has a cake to deliver. Every mile along the highway, in both directions, there is a lamppost. You want to program the robot to move exactly **N** lampposts to the east, and release the cake there. The route does not have to be direct, as long as the robot eventually releases the cake in the right place.

Unfortunately, the robot comes equipped with only very little memory, and it is capable of no advanced logic. To control the robot you will have to give it a very simple program at the start that will get it to release the cake at the proper location. This program must be composed of one or more statements, each of which tells the robot what to do under certain conditions. These statements must be in the following format:

`<S> <M> -> <action>`

which means that if all of the following conditions are met:

1. The robot is in state `S`.
2. The robot is at a lightpost marked with number `M`.

then it will perform exactly one of the following actions:

1. Mark the current post with a new number, change state and move. To do this `<action>` must be formatted as "`<D> <NS> <NM>`", where `D` is the direction to move (use 'W' for west and 'E' for east), `NS` is the robot's new state and `NM` is the new mark for the current lightpost.
2. Release the cake at the current position and self-destruct. To do this `<action>` must be formatted as "`R`".

If you output two or more statements with the same values of `S` and `M`, the robot will misbehave and destroy the cake.

If at any time the robot is in a state X at a lamppost marked with Y such that there is no statement with `S=X` and `M=Y`, then the robot will get confused and eat the cake.

All states and marks must be integers with absolute value no greater than one million (10^6). Assume that initially the robot is in state zero and all lampposts are marked with zero.

Given **N**, write a program so the robot releases the cake in the appropriate place. Your program must use at most 30 statements, and it must terminate within **X** steps.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of a single line containing an integer **N**, which indicates the lamppost where the robot must release the cake.

Output

For each test case, first output "Case #x: y", where x is the number of the test case (starting with 1) and y is the number of statements you will use. Next output y lines, each of which represents a statement for the robot in the format described previously.

WARNING: Judge's response might take up to 5 seconds longer than usual to appear, because your output is run as part of the validation.

Limits

1 ≤ **T** ≤ 15.
Memory limit: 1GB.

Small dataset (Test set 1 - Visible)

0 ≤ **N** ≤ 500.
X = 250,000 (2.5×10^5).
Time limit: 30 seconds.

Large dataset (Test set 2 - Hidden)

0 ≤ **N** ≤ 5000.
X = 150,000 (1.5×10^5)
Time limit: 60 seconds.

Sample

Input Output

```
    Case #1: 1
    0 0 -> R
    Case #2: 5
    0 0 -> E 1 1
3   1 0 -> E 2 1
0   2 0 -> E 3 1
4   3 0 -> E -1 1
0   -1 0 -> R
    Case #3: 3
    0 0 -> E 1 1
    0 1 -> R
    1 0 -> W 0 1
```

In the first case, the robot is initially in state zero, and there is a zero on the lamppost. So it executes its only statement, which is to release the cake.

In the second case, the robot has five states: 0, 1, 2, 3, and -1. The robot performs the following actions:

- Mark the current lamppost with a 1, move east, and go to state 1.
- Mark the current lamppost with a 1, move east, and go to state 2.
- Mark the current lamppost with a 1, move east, and go to state 3.
- Mark the current lamppost with a 1, move east, and go to state -1.
- Release the cake.

In the third case, the robot has two states, and performs the following actions:

- Mark the current lamppost with a 1, move east, and go to state 1.
- Mark the current lamppost with a 1, move west, and go to state 0.
- Release the cake.

Note that the robot takes different actions at the two times it is in state 0, because it sees a different mark each time.