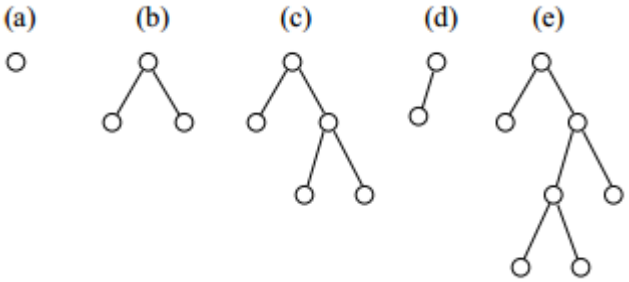
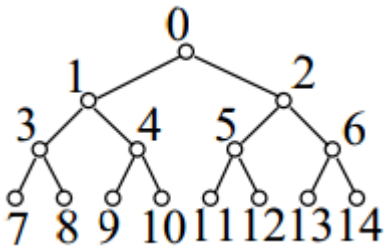
Right Trees

Program Name: `RightTree.java` **Input File:** `righttree.dat`

A right-tree is a binary tree that has the following property: for every node in the tree, the right subtree has at least as many nodes as the left subtree. Thus, of the following (a), (b) and (c) are right-trees while (d) and (e) are not.



The binary trees are encoded using a string of characters. The node positions in a binary tree can be numbered from top to bottom, left to right starting with Position 0 for the root node. The figure below shows the position numbers for a tree of height 4. The scheme continues on the next row (with nodes 15, 16, ...) for larger trees.



Input

Let *s* be the binary string that encodes a tree where s_i is the character at position *i*. The characters in *s* are numbered left to right starting at 0 so that s_0 is the left-most character in the string. If $s_i = 1$ then there is a node at position *i* in the tree, else $s_i = 0$ indicates no node. The string terminates with the last "1". All trees given as input here will have height no more than 8. Also assume the strings are syntactically correct in that if a "0" indicates no node at some point, there can never be any "1"s in positions that are nodes on that "0"s subtree.

The input consists of an integer *n* followed by *n* test cases, one per line. Each test case consists of a string encoded binary tree.

Output

For each problem instance, report one of the following messages corresponding to whether the tree is a right-tree or not:

```
Tree n is a right-tree.
Tree n is not a right-tree.
```

accordingly, where *n* is the test case number, starting at 1.

Sample Input

```
5
1
111
1110011
11
1110011000011
```

Sample Output

```
Tree 1 is a right-tree.
Tree 2 is a right-tree.
Tree 3 is a right-tree.
Tree 4 is not a right-tree.
Tree 5 is not a right-tree.
```