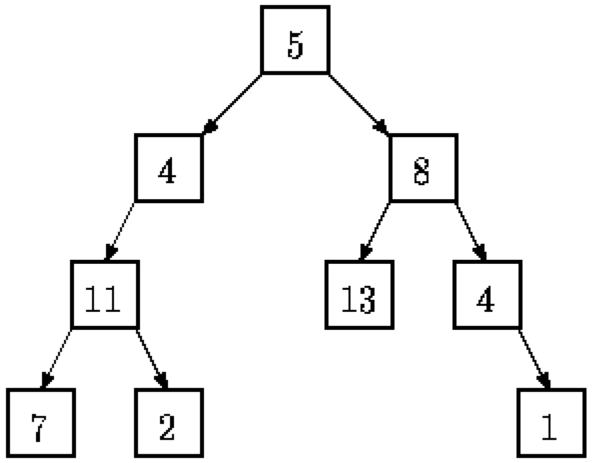
Lab02 Binary Tree Construction Based on Path Encoding

The Problem

Given a path description of a binary tree, you are to write a program that prints a level-order traversal of the tree. For example, a level order traversal of the tree below



is: 5, 4, 8, 11, 13, 4, 7, 2, 1.

Each node is specified by **(*n*,*s*)** where *n* is the value at the node whose path from the root is given by the string *s*. A path is a sequence of *L*'s and *R*'s where *L* indicates a left branch and *R* indicates a right branch. In the tree above, the node containing 13 is specified by (13,RL), and the node containing 2 is specified by (2,LLR). The root node is specified by (5,) where the path is an empty string.

A binary tree is considered to be *completely specified* if every node on all root-to-node paths in the tree is given a value.

The Input

The input is a tree described in a sequence of pairs (*n*,*s*) separated by whitespace. The last pair is (). No whitespace appears between left and right parentheses.

All nodes contain positive integers. Every tree in the input will consist of at least one node and no more than 256 nodes.

The Output

Print the level order traversal of the binary tree specified. If a tree is not completely specified, i.e. some node in the tree is NOT given a value, then the string "not complete" should be printed before printing the level order traversal.

Sample Input 1

```
(11,LL) (7,LLL) (8,R)
(5,) (4,L) (13,RL) (2,LLR) (1,RRR) (4,RR) ()
```

Sample Output 1

```
levelorder:5 4 8 11 13 4 7 2 1
```

Sample Input 2

```
(3,L) (4,R) ()
```

Sample Output 2

```
not complete
levelorder:-1 3 4
```

```
Note that -1 is used to indicate that the value has not been specified in
sample input 2.
```

Program Development

The given test driver TestBinaryTree.java can be modified in order to meet the output requirement. The class NP is given so that you can use it to store path encoding information when needed. Node.java is the Node class needed by the BinaryTree class. You are not required to modify them.

You are to implement readBinaryTree1(BufferedReader bR) in the file BinaryTree.java to read the path encodings and create the nodes of the binary tree.

You should create at least 5 sets of test data for a thorough testing. The test data files are stored in the **in** folder and each file is named as in1.txt, in2.txt, … . Similarly, the outputs should be stored in the **out** folder and each file is named correspondingly as out1.txt, out2.txt, … .

In the solution folder, in addition to all the source files, create the file **testHarness.txt** to describe what you want to test with each of the test data files. For example,

> in1.txt:
> The given sample input 1. It is used to test whether the sample output 1 can be produced.
> in2.txt:
> The given sample input 2. It is used to test whether the sample output 2 can be produced.
> in3.txt:
> It is used to test whether an empty binary tree can be produced.
> in4.txt:
> …

Submission

Zip the complete package that includes the solution folder of all the source files and the testHarness.txt file, the in folder of all test data, and the out folder of the corresponding output data, name it ***Lab02g<YourLabGroupNo><YourMatricNo>.zip***. Submit the zip file into the correct folder in your group's workbin.