YONSEI UNIVERSITY

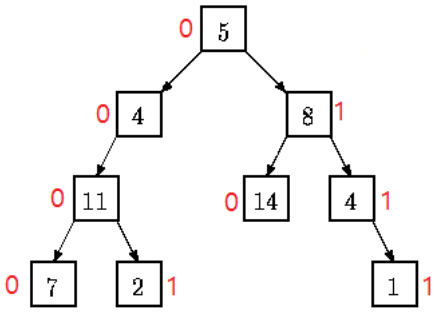
CSI3108-01 2015. 09. 02

Programming HW#1

Max 20 points

Due on Sept. 8 (Tuesday), 2015, by 5 pm

Given a binary tree of nodes each of which contains a positive integer, you are to write a Java program that determines whether there exists a root-to-leaf path whose nodes sum to a specified integer. For example, in the tree shown below there are exactly four root-to-leaf paths. The sums of the paths are 27, 22, 27, and 18.



Binary trees are represented in the input file having the following form.

empty tree ::= ()
tree ::= *empty tree* | (**int** *tree tree*)

The tree diagrammed above is represented by the expression

(5 (4 (11 (7 () ()) (2 () ())) ()) (8 (13 () ()) (4 () (1 () ()))))

Note that with this formulation all leaves of a tree are of the form **(int () ())**

Input

The input consists of a sequence of 20 test cases, one line per test case. Each test case consists of an integer ***n*** followed by a binary tree formatted as described above, where ***n*** can be up to 1,000. Assume all input expressions are valid.

Output

There should be one line of output for each test case in the input file. For each test case, the output is the string of 0's and 1's if there is a root-to-leaf path in the tree whose sum is ***n***, where when you follow the path from the root if you move down to the left add '0' to the string and '1' otherwise. Note that the root is '0'. In case when there are multiple paths with the same summation ***n***, print the path that is lexicographically first. Print '-1' if there is no path in the tree whose sum is ***n***.

Sample Input (c:\hw1\input.txt)

```
20                                  // the number of test cases
22 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))      // test case #1
27 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))      // test case #2
18 (5(4(11(7()())(2()()))()) (8(13()())(4()(1()()))))      // …
15 (3(2 (4 () () )(8 () () ) ) (1 (6 () () )(4 () () ) ) )
5 ()
...                                 // omitted test case up to #20
```

Sample Output (c:\hw1*studentID*.txt)

```
0001                    // test case #1
0000                    // test case #2
0111                    // …
-1
-1
...
```