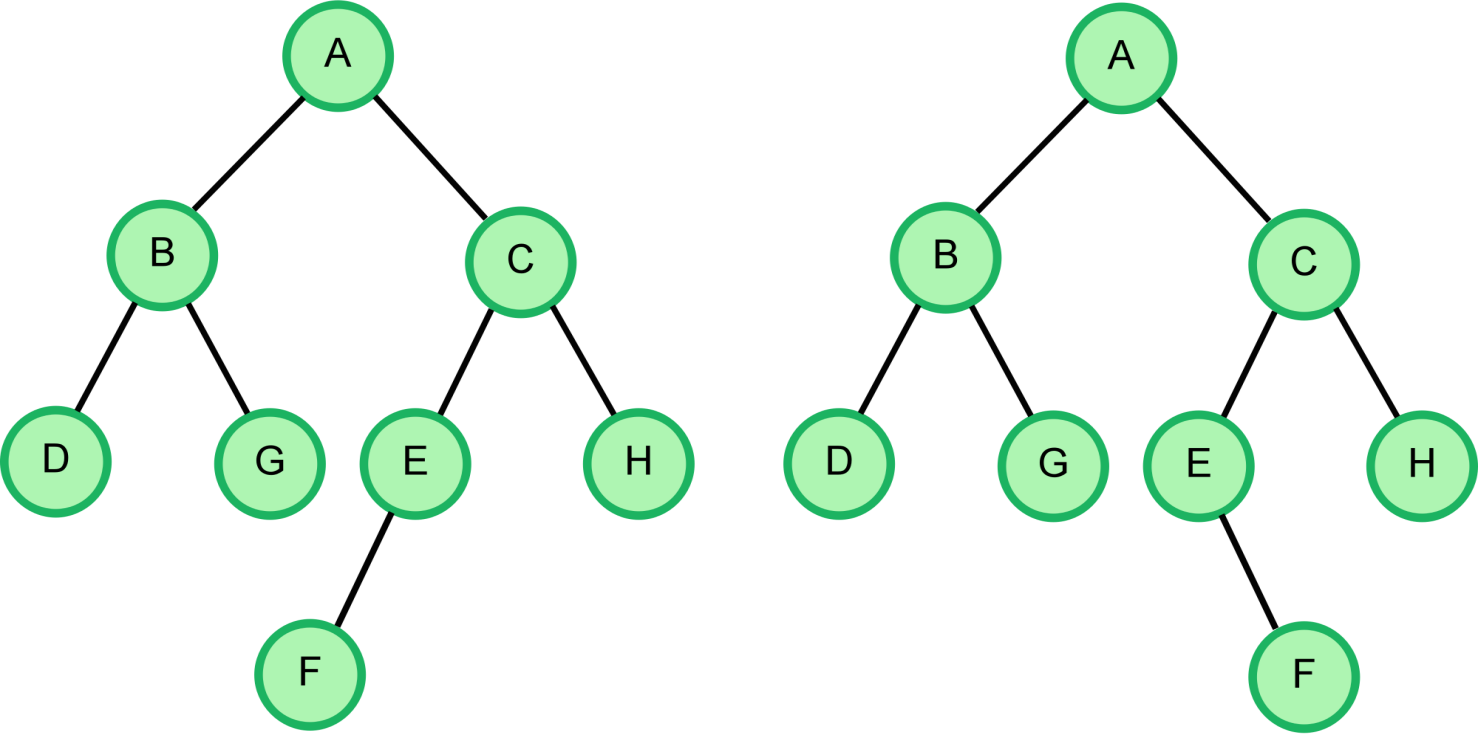
Is this a tree?

A binary tree is represented as a sequence of parent-child pairs, for example:

(A,B) (A,C) (B,G) (C,H) (E,F) (B,D) (C,E)

A tree with those edges may be illustrated in many ways. Here are two:



The following is a recursive definition for the S-expression of a tree:

S-exp(node) = ( node->val (S-exp(node->first\_child))(S-exp(node->second\_child))), if node != NULL = "", node == NULL

where, first\_child->val < second\_child->val (first\_child->val is lexicographically smaller than second\_child-> val)

This tree can be represented in an S-expression in multiple ways. The lexicographically smallest way of expressing it is as follows:

(A(B(D)(G))(C(E(F))(H)))

Translate the node-pair representation into its lexicographically smallest S-expression or report any errors that do not conform to the definition of a binary tree.

The list of errors with their codes is as follows:

****Error Code      Type of error****  
E1                 More than 2 children  
E2                 Duplicate Edges  
E3                 Cycle present (node is direct descendant of more than one node)  
E4                 Multiple roots  
E5                 Any other error

****Function Description****

Complete the function *sExpression* in the editor below. The function must return either the lexicographically lowest S-expression or the lexicographically lowest error code as a string.

sExpression has the following parameter(s):

*nodes:*  a string of space-separated parenthetical elements, each of which contains the names of two connected nodes separated by a comma

****Constraints:****

1. All node names are single characters in the range *ascii[A-Z]*
2. The maximum node count is *26*.
3. There is no specific order to the input (parent, child) pairs.

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains a string *nodes*.

Sample Case 0

****Sample Input 0****

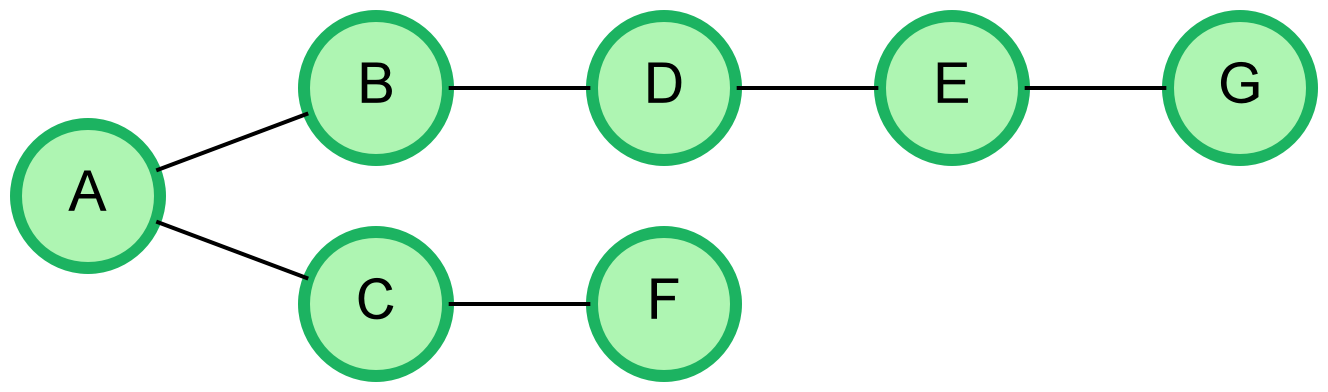
(B,D) (D,E) (A,B) (C,F) (E,G) (A,C)

****Sample Output 0****

(A(B(D(E(G))))(C(F)))

****Explanation 0****

A representation of the tree is as follows:



Sample Case 1

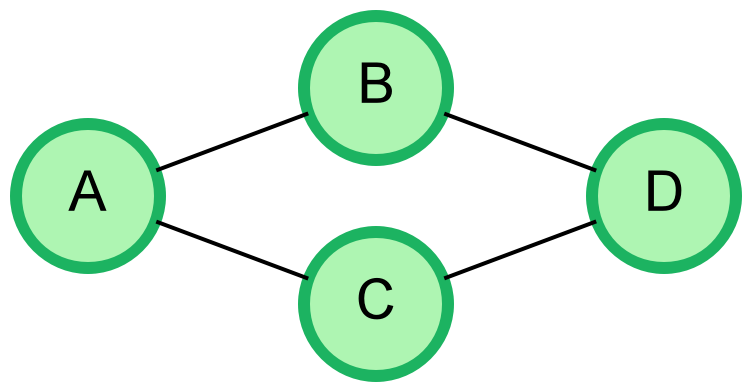
****Sample Input 1****

(A,B) (A,C) (B,D) (D,C)

****Sample Output 1****

E3

****Explanation 1****



Node *C*is a child of nodes *A* and *D.* Since *D* tries to attach itself as parent to a node already above it in the tree, this forms a cycle.

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'sExpression' function below.

\*

\* The function is expected to return a STRING.

\* The function accepts STRING nodes as parameter.

\*/

public static String sExpression(String nodes) {

// Write your code here

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String nodes = bufferedReader.readLine();

String result = Result.sExpression(nodes);

bufferedWriter.write(result);

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*