The following us 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 (lexicographically smaller)
```

This tree can be represented in a S-expression in multiple ways. The lexicographically smallest way of expressing this 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
E4	Multiple roots
E5	Any other error

☆ Maximizing Profit from Stocks

Your algorithms have become so good at predicting the market that you now know what the share price of Silly Purple Toothpicks Inc. (SPT) will be for a number of minutes going forward. Each minute, your high frequency trading platform allows you to either buy one share of SPT, sell any number of shares of SPT that you own, or not make any transaction at all. Your task is to find the maximum profit you can obtain with an optimal trading strategy.

Given a date string in the format *Day Month Year*, where:

- *Day* is in the set *{"1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th", "9th", …, "29th", "30th", "31st"}*.
- *Month* is in the set *{"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}*.
- *Year* is in the inclusive range *[1900, 2100]*.

Convert the date string to the format *YYYY-MM-DD*, where:

- *YYYY* denotes the *4* digit year.
- *MM* denotes the *2* digit month.
- *DD* denotes the *2* digit day.

For example:

- *1st Mar 1984 → 1984-03-01*
- *2nd Feb 2013 → 2013-02-02*
- *4th Apr 1900 → 1900-04-04*