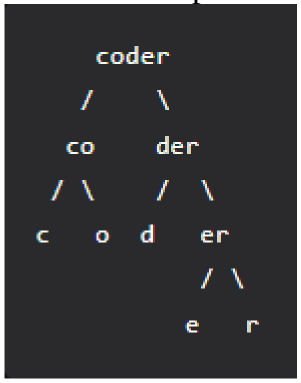
Scrambled String

Given two strings **S1** and **S2** of equal length, the task is to determine if S2 is a scrambled form of S1.

Scrambled string: Given string **str**, we can represent it as a binary tree by partitioning it into two non-empty substrings recursively.
Below is one possible representation of str = **coder:**



```
      coder
     /    \
    co     der
   / \    /  \
  c   o  d   er
            / \
           e   r
```

To scramble the string, we may choose any non-leaf node and swap its two children. Suppose, we choose the node **co** and swap its two children, it produces a scrambled string **ocder**.
Similarly, if we continue to swap the children of nodes **der** and **er**, it produces a scrambled string **ocred**.

Note: Scrambled string is not the same as an Anagram.

Print "Yes" if S2 is a scrambled form of S1 otherwise print "No".

Example 1:

```
Input: S1="coder", S2="ocder"
Output: Yes
Explanation: ocder is a scrambled
form of coder.

    ocder
   /     \
  oc    der
 / \
o   c

As "ocder" can represent it
as a binary tree by partitioning
it into two non-empty substrings.
We just have to swap 'o' and 'c'
to get "coder".
```

Example 2:

```
Input: S1="abcde", S2="caebd"
Output: No
Explanation: caebd is not a
scrambled form of abcde.
```

Your Task:

You don't need to read input or print anything. You only need to complete the function **isScramble()** which takes two strings S1 and S2 as input and returns a boolean value.

Expected Time Complexity: O(N^2)
Expected Auxiliary Space: O(N^2)

Constrains:

- S1.length = S2.length
- S1.length<=31
- S1 and S2 consist of lower-case English letters.