A valid parentheses string is either empty (""), "(" + A + ")", or A + B, where A and B are valid parentheses strings, and + represents string concatenation.  For example, "", "()", "(())()", and "(()(()))" are all valid parentheses strings.

A valid parentheses string S is **primitive** if it is nonempty, and there does not exist a way to split it into S = A+B, with A and B nonempty valid parentheses strings.

Given a valid parentheses string S, consider its primitive decomposition: S = P\_1 + P\_2 + ... + P\_k, where P\_i are primitive valid parentheses strings.

Return S after removing the outermost parentheses of every primitive string in the primitive decomposition of S.

**Example 1:**

**Input:** "(()())(())"

**Output:** "()()()"

**Explanation:**

The input string is "(()())(())", with primitive decomposition "(()())" + "(())".

After removing outer parentheses of each part, this is "()()" + "()" = "()()()".

**Example 2:**

**Input:** "(()())(())(()(()))"

**Output:** "()()()()(())"

**Explanation:**

The input string is "(()())(())(()(()))", with primitive decomposition "(()())" + "(())" + "(()(()))".

After removing outer parentheses of each part, this is "()()" + "()" + "()(())" = "()()()()(())".

**Example 3:**

**Input:** "()()"

**Output:** ""

**Explanation:**

The input string is "()()", with primitive decomposition "()" + "()".

After removing outer parentheses of each part, this is "" + "" = "".

**Note:**

1. S.length <= 10000
2. S[i] is "(" or ")"
3. S is a valid parentheses string