# 301. Remove Invalid Parentheses

Given a string `s` that contains parentheses and letters, remove the minimum number of invalid parentheses to make the input string valid.

Return *all the possible results*. You may return the answer in **any order**.

**Example 1:**

```
Input: s = "()())()"
Output: ["(())()","()()()"]
```

**Example 2:**

```
Input: s = "(a)())()"
Output: ["(a())()","(a)()()"]
```

**Example 3:**

```
Input: s = ")("
Output: [""]
```

```python
class Solution:
    def removeInvalidParentheses(self, s: str) -> List[str]:
        def num_invalid(s):
            num_left = num_right = 0
            for char in s:
                if char == "(":
                    num_left += 1
                elif char == ")":
                    if num_left > 0:
                        num_left -= 1
                    else:
                        num_right += 1
            return [num_left, num_right]

        valid = [0, 0]
        # rem_left and rem_right represent the number of left and right parentheses we are allowed to remove
        # to meet the requirement of only removing the minimum amount of invalid parentheses
```

```
        def backtrack(s, ind, size, rem_left, rem_right, result):
            if rem_left == 0 and rem_right == 0:
                if num_invalid(s) == valid:
                    result.append(s)
                return

            for i in range(ind, size):
                current = s[i]
                if i > ind and current == s[i - 1]:
                    continue

                if current == ')' and rem_right > 0:
                    backtrack(s[:i] + s[i + 1:], i, size - 1, rem_left,
rem_right - 1, result)
                elif current == '(' and rem_left > 0:
                    backtrack(s[:i] + s[i + 1:], i, size - 1, rem_left -
1, rem_right, result)

        l, r = num_invalid(s)
        if l == 0 and r == 0:
            return [s]

        result = []
        backtrack(s, 0, len(s), l, r, result)
        return result
```