Given a string s of '(' , ')' and lowercase English characters.

Your task is to remove the minimum number of parentheses ( '(' or ')', in any positions ) so that the resulting *parentheses string* is valid and return **any** valid string.

Formally, a *parentheses string* is valid if and only if:

* It is the empty string, contains only lowercase characters, or
* It can be written as AB (A concatenated with B), where A and B are valid strings, or
* It can be written as (A), where A is a valid string.

**Example 1:**

**Input:** s = "lee(t(c)o)de)"

**Output:** "lee(t(c)o)de"

**Explanation:** "lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.

**Example 2:**

**Input:** s = "a)b(c)d"

**Output:** "ab(c)d"

**Example 3:**

**Input:** s = "))(("

**Output:** ""

**Explanation:** An empty string is also valid.

**Constraints:**

* 1 <= s.length <= 105
* s[i] is either'(' , ')', or lowercase English letter.

Solution:

class Solution {

public:

    string minRemoveToMakeValid(string s) {

        int totalOpenP = 0, totalCloseP = 0;

        for(auto i : s){

            if(i == '('){

                totalOpenP++;

            }

            else if(i == ')'){

                totalCloseP++;

            }

        }

        int currOpenP = 0, currCloseP = 0;

        string newS;

        for(auto i : s){

            if(i == '('){

                currOpenP++;

            }

            else if(i == ')'){

                currCloseP++;

            }

            if(i != '(' and i != ')'){

                newS += i;

            }

            else{

                if(i == '(' && totalCloseP >= currOpenP){

                    newS += i;

                }

                if(i == ')' && currOpenP > 0){

                    newS += i;

                    currOpenP--;

                    totalCloseP--;

                }

                else if(i == ')'){

                    totalCloseP--;

                }

            }

        }

        return newS;

    }

};