4) REMOVE INVALID PARENTHESES:

METHOD:(THERE IS GLITCH IN SUM ....THE EFFICIENT TECHNIQUE SHOWS TLE)
LINK OF EXPLANATION : Remove Invalid Parentheses Explained using Stacks | Leetcode 3..

1) The First thing we need in this question is to balance the string.
   So in order to check whether the given string is balanced or not, we need a function for that.
   So write a function to check the balancing of the parenthesis.

```
bool isBalance(string &s){
        stack<char> st;
        for(int i=0;i<s.length();i++){
            if(s[i]=='('){
                st.push(s[i]);
            }
            else if(s[i]==')'){
                if(st.size()!=0){
                    st.pop();
                }
                else{
                    return false;
                }
            }
        }
        if(st.size()!=0){
            return false;
        }
        return true;
    }
```

2) The second thing in the question is that , we can only remove the parentheses which makes the original string unbalanced.
   So we have to count the number of such parentheses which can be done in a very similar way which we use for checking the balancing of the parentheses.

```
int countInvalid(string &s){
        stack<char> st;
        int count=0;
        for(int i=0;i<s.length();i++){
            if(s[i]=='('){
                st.push(s[i]);
            }
            else if(s[i]==')'){
                if(st.size()!=0 && st.top()=='('){
                    st.pop();
                }
                else{
                    count++;
                }
            }
        }
        while(st.size()!=0){
            count++;
            st.pop();
        }
        return count;
    }
```

3) Now the only thing left is the main recursion process.
   The method used by me showed TLE.My method was based on selection method i.e in one recursion we select a particular character and in other recursion we left this character and move on to the other character.

```
void fun(int index,string &t,int count,string &s,vector<string> &answer){
        if(index==s.length()){
            if(isBalance(t)){
                for(auto it=answer.begin();it!=answer.end();it++){
                    if(*it==t){
                        return;
                    }
                }
                answer.push_back(t);
                return;
            }
        }
        else{
            if(count==0){
                t=t+s[index];
                fun(index+1,t,count,s,answer);
                t.pop_back();
            }
            else{
                t=t+s[index];
                fun(index+1,t,count,s,answer);
                t.pop_back();
                fun(index+1,t,count-1,s,answer);
            }
        }
    }
```

The other method which also shows TLE:

```
void fun(string s,int count,vector<string> &answer,unordered_map<string,int> &m){
        if(count==0){
            if(isBalance(s)==true && m[s]==0){
                m[s]=1;
                answer.push_back(s);
            }
            return;
        }
        else{
            for(int i=0;i<s.length();i++){
```

```
            string t=s.substr(0,i)+s.substr(i+1);
            fun(t,count-1,answer,m);
        }
    }
}
vector<string> removeInvalidParentheses(string s) {
    int count=countInvalid(s);
    vector<string> answer;
    unordered_map<string,int> m;
    fun(s,count,answer,m);
    return answer;
}
```