Longest Valid Parentheses

Given a string S consisting of opening and closing parenthesis '(' and ')'. Find length of the longest valid parentheses substring.

Input: ((() **Output:** 2

Input:)()()) **Output:** 4

Algorithm:

1. Create an empty stack and push -1 to it. The first element of the stack is used to provide a base for the next valid string.
2. Initialize the result as 0.
3. If the character is '(' i.e. str[i] == '(', push index 'i' to the stack.
4. Else (if the character is ')')
   - Pop an item from the stack (Most of the time an opening bracket)
   - If the stack is not empty, then find the length of current valid substring by taking the difference between the current index and top of the stack. If current length is more than the result , then update the result.
   - If the stack is empty, push the current index as a base for the next valid substring.
5. Return result.

Code:

```
#include <bits/stdc++.h>
using namespace std;


int main(){
    int t ;
    cin >> t;
    while(t--){
```

```
        string s;
        cin >> s;
        stack<int> st;
        st.push(-1);
        int ans =0;
        for(int i=0;i<s.length();i++){
            if(s[i]=='(')
            st.push(i);
            else{
                st.pop();
                if (!st.empty()) ans = max(ans,i - st.top());
                else{
                    st.push(i);
                }

            }
        }
        cout << ans << endl;
    }

}
```