HW_Minimum Add to Make Parentheses Valid 3

A parentheses string is valid if and only if:

It is the empty string, 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. You are given a parentheses string s. In one move, you can insert a parenthesis at any position of the string.

For example, if s = "())", you can insert an opening parenthesis to be "(())" or a closing parenthesis to be "())(". Return the minimum number of moves required to make s valid.

Input Format

First line contains a string s.

Constraints

1 <= s.length <= 1000 s[i] is either '(' or ')'.

Output Format

Returns an integer value.

Sample Input 0

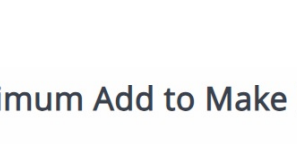
```
())
```

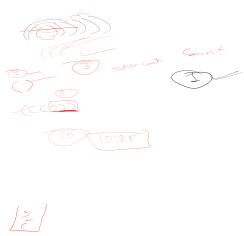
Sample Output 0

```
1
```

Explanation 0

Only 1 opening bracket will make the string s valid "(())"





```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();

        Stack<Character> st = new Stack<>();
        for(int i =0;i<str.length();i++){
            char ch = str.charAt(i);

            if(!st.isEmpty() && ch==')' && st.peek()=='(')st.pop();
            else st.push(ch);
        }
        System.out.print(st.size());
    }
}
```

HW_Next greater element on left 1

Given an array arr, print the **Next Greater Element** (NGE) for every element. The Next greater Element for an element x is the first greater element on the left side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Input Format

- First line contains an integer N representing the size of the array.
- Next N lines contains elements of the array.

Constraints

```
1 <= N <= 1000
-1000 <= arr[i] <= 1000
```

Output Format

- Print the answer array seperated by a single space

Sample Input 0

```
5
3 4 3 2 1
```

Sample Output 0

```
-1 -1 4 3 2
```

Explanation 0

- No element on left of $arr[0]$ = 3, which is greater so $res[0]$ = -1.
- No element on left of $arr[1]$ = 4, which is greater so $res[1]$ = -1.
- 4 is nearest element to the left of $arr[2]$ = 3, which is greater so $res[2]$ = 4.
- 3 is nearest element to the left of $arr[3]$ = 2, which is greater so $res[3]$ = 3.
- 2 is nearest element to the left of $arr[4]$ = 1, which is greater so $res[2]$ = 2.

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n  = sc.nextInt();
        Stack<Integer> st = new Stack<>();
        for(int i=0;i<n;i++){
            int val = sc.nextInt();

            while(!st.isEmpty() && st.peek()<=val)st.pop();
            if(st.isEmpty())System.out.print(-1+" ");
            else System.out.print(st.peek()+" ");
            st.push(val);
        }
    }
}
```