.split('t')

vikash ____ 's

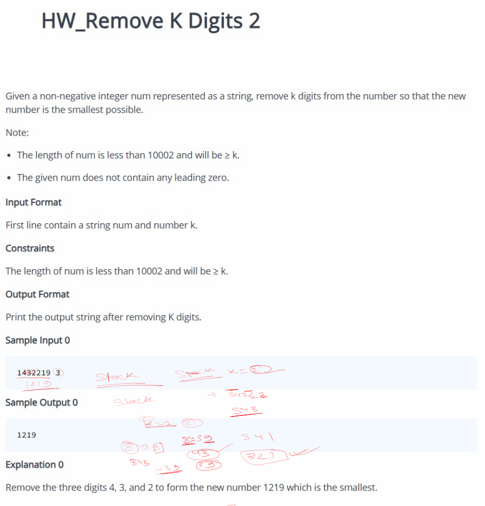
str = vikash's a good boy

array = str.split('list')

t

vikash ______'s

vikash t ~~'s~~



HW_Remove K Digits 2

Given a non-negative integer num represented as a string, remove k digits from the number so that the new number is the smallest possible.

Note:

- The length of num is less than 10002 and will be ≥ k.
- The given num does not contain any leading zero.

Input Format

First line contain a string num and number k.

Constraints

The length of num is less than 10002 and will be ≥ k.

Output Format

Print the output string after removing K digits.

Sample Input 0

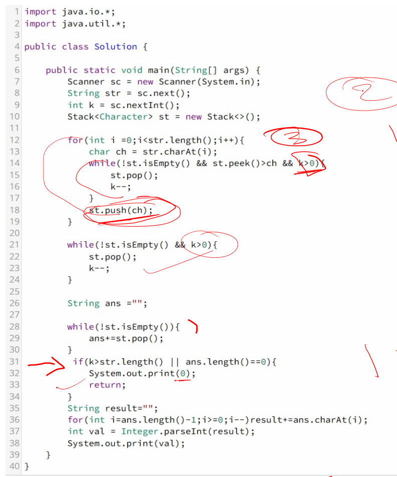
```
1432219 3
```

Sample Output 0

```
1219
```

Explanation 0

Remove the three digits 4, 3, and 2 to form the new number 1219 which is the smallest.



```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.next();
        int k = sc.nextInt();
        Stack<Character> st = new Stack<>();

        for(int i =0;i<str.length();i++){
            char ch = str.charAt(i);
            while(!st.isEmpty() && st.peek()>ch && k>0){
                st.pop();
                k--;
            }
            st.push(ch);
        }

        while(!st.isEmpty() && k>0){
            st.pop();
            k--;
        }

        String ans ="";

        while(!st.isEmpty()){
            ans+=st.pop();
        }
         if(k>str.length() || ans.length()==0){
            System.out.print(0);
            return;
        }
        String result="";
        for(int i=ans.length()-1;i>=0;i--)result+=ans.charAt(i);
        int val = Integer.parseInt(result);
        System.out.print(val);
    }
}
```

k = 4

ans = 0

k = 3

193221 9

13221 9

12219

for(i=0; i<s.length; i++){

while(st.peek() > str.charAt(i))

st.pop();

k--;

st.push(ch);

1432219

ParseInt

= empty

k = 5

integer = empty

HW_Remove Outermost Parentheses 4

A valid parentheses string is either empty "", "(" + A + ")", or A + B, where A and B are valid parentheses strings, and + represents string concatenation.

Return s after removing the outermost parentheses of every primitive string in the primitive decomposition of s.

Input Format

The first line be String S .

Constraints

1 <= s.length <= 10^5

s[i] is either '(' or ')'.

s is a valid parentheses string.

Output Format

Return s after removing the outermost parentheses of every primitive string in the primitive decomposition of s.

Sample Input 0

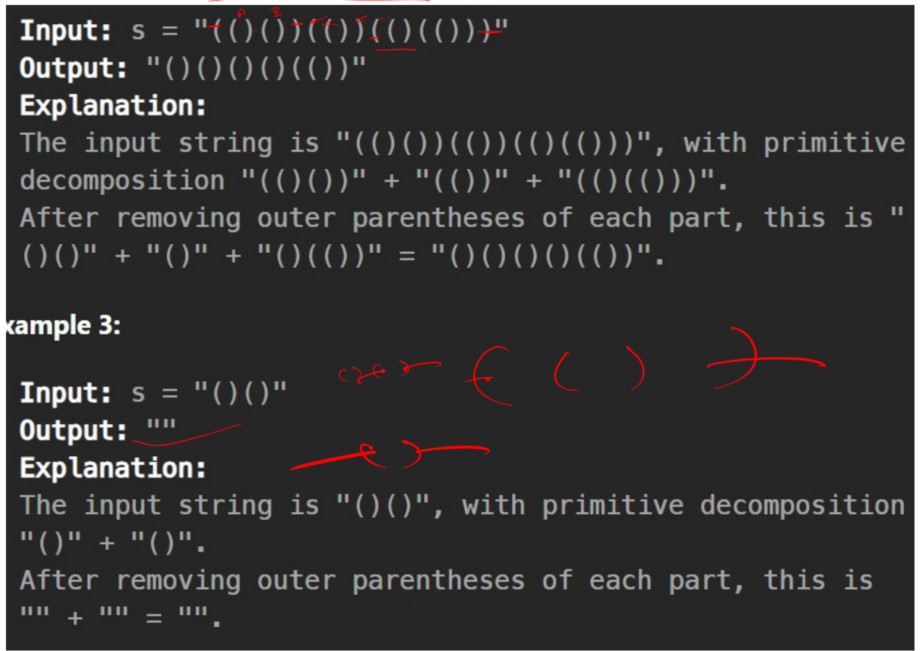
```
(()())(())
```

Sample Output 0

```
()()()
```

empty

output



```
Input: s = "(()())(())(()(()))"
Output: "()()()()(())"
Explanation:
The input string is "(()())(())(()(()))", with primitive
decomposition "(()())" + "(())" + "(()(()))".
After removing outer parentheses of each part, this is "
()()" + "()" + "()(())" = "()()()()(())".
```

xample 3:

```
Input: s = "()()"
Output: ""
Explanation:
The input string is "()()", with primitive decomposition
"()" + "()".
After removing outer parentheses of each part, this is
"" + "" = "".
```