- **MEDIUM**
- **Max Score: 40 Points**
-
-

Remove All Adjacent Duplicates In String

You are given a string s consisting of lowercase English letters. A duplicate removal consists of choosing two adjacent and equal letters and removing them.

We repeatedly make duplicate removals on s until we no longer can.

Print the final string after all such duplicate removals have been made. It can be proven that the answer is unique.

Your task is to complete the function `removeDuplicates` which receives the input string and returns the final string after all duplicates have been removed.

Input Format

The first line contains the string s.

Output Format

Print the final string after all removals.

Example 1

Input

```
abbaca
```

Output

```
ca
```

Explanation

For example, in "abbaca" we could remove "bb" since the letters are adjacent and equal, and this is the only possible move. The result of this move is that the string is "aaca", of which only "aa" is possible, so the final string is "ca".

Example 2

Input

```
azxxzy
```

Output

```
ay
```

Explanation

For example, in "azxxzy" we could remove "xx" since the letters are adjacent and equal, and this is the only possible move. The result of this move is that the string is "azzy", of which only "zz" is possible, so the final string is "ay".

Constraints

1 <= s.length <= 10^5

s consists of lowercase English letters.

Topic Tags

- **Strings**
- **Loops**
- **Arrays**

My code

```java
// n java
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main
{
    public static void main (String[] args) throws
java.lang.Exception
    {
        //your code here
    //this is question of stack
        Scanner s=new Scanner(System.in);
        String str=s.next();
        char stack[]=new char[str.length()];
        int top=-1;
        for(int i=0;i<str.length();i++)
            {
                if(top==-1)
                    stack[++top]=str.charAt(i);
                else
                {
                  if( stack[top]==str.charAt(i))
                      top--;
                  else
                      stack[++top]=str.charAt(i);
                }
```

```java
        }
    for(int i=0;i<=top;i++)
        System.out.print(stack[i]);
    }
}
```