

Problem: Reduced String

Steve has a string, `s`, consisting of lowercase English alphabetic letters. In one operation, he can delete any *pair of adjacent letters* with same value. For example, string "aabcc" would become either "aab" or "bcc" after operation.

Steve wants to reduce `s` as much as possible. To do this, he will repeat the above operation as many times as it can be performed. Help Steve out by finding and printing `s`'s non-reducible form!

Note: If the final string is empty, print `Empty String`.

Input Format

A single string, `s`.

Constraints

-

Output Format

If the final string is empty, print `Empty String`; otherwise, print the final non-reducible string.

Sample Input 0

```
aaabccddd
```

Sample Output 0

```
abd
```

Sample Case 0

Steve can perform the following sequence of operations to get the final string:

1. `aaabccddd` → `abccddd`
2. `abccddd` → `abddd`
3. `abddd` → `abd`

Thus, we print `abd`.

Sample Input 1

```
baab
```

Sample Output 1

```
Empty String
```

Explanation 1

Steve can perform the following sequence of operations to get the final string:

1. `baab` → `bb`
2. `bb` → `Empty String`

Thus, we print `Empty String`.

Sample Input 2

aa

Sample Output 2

Empty String

Explanation 2

Steve can perform the following sequence of operations to get the final string:

1. aa → Empty String

Thus, we print Empty String.

Solution

```
int main()
{
    string str;
    cin >> str;
    int length = str.length();
    for(int i=0; i<length-1; i++)
    {
        if(str[i] == str[i+1])
            /*Perform array shifting*/
            for(int j=i; j<length-2; j++)
                {str[j] = str[j+2];}
            length -= 2;
            i = -1;
        }
    }
    if(length == 0){cout << "Empty String"; return 0;}
    for(int i=0; i<length; i++)
        {cout << str[i];}
    return 0;
}
```

‘Anshul Aggarwal