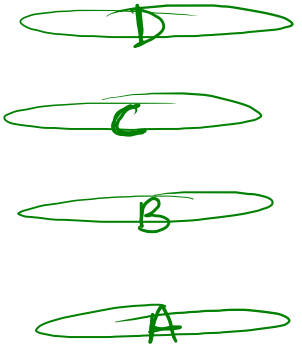
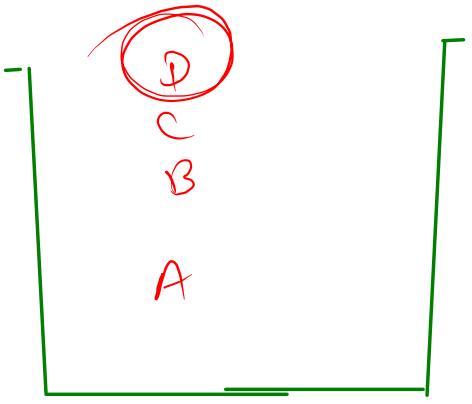
Stack

↳ kind of bucket

remove



get → top → access



idx → not accessible

LIFO : Last in first out

Stack:

initialize

add

remove

get

size

```java
import java.util.Stack;
import java.util.*;

public class Main
{
    public static void main(String[] args) {
        //init
        //ArrayList<Integer> arr = new ArrayList<>();
        Stack<Integer> st = new Stack<>();
        //add: push
        st.push(10);
        st.push(20);
        st.push(30);

        System.out.println(st.peek());
        st.push(40);

        //remove: pop
        System.out.println(st.pop());

        System.out.println(st.size());


```

```
30
```

Stack Syntax Learning

1. **Declare an Empty** $stack\ s$.
2. Take Single Integer T as input.
3. For next T Lines format $(case, x(optional))$

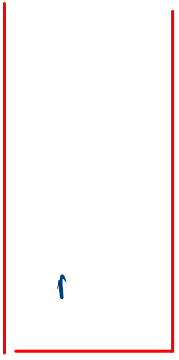
- case 1. $Print$ the $size$ of the $stack$ in a separate line.
- case 2. $Remove$ an element from the stack. If the stack is empty then print -1 in a separate line.
- case 3. Add Integer x to the $stack\ s$.
- case 4. $Print$ an element at the top of the $stack$. If stack is empty print -1 in a seperate line.

Sample Input 0

```
10
3 1
3 2
4
4
2
4
3 4
2
4
1
```

Sample Output 0

```
2
2
1
1
-1
```



```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int t = scn.nextInt();
        Stack<Integer> st = new Stack();

        for(int i = 1; i <= t; i++){
            int caseNu = scn.nextInt();
            if(caseNu == 1){
                System.out.println(st.size());
            }else if(caseNu == 2){
                if(st.size()==0){
                    System.out.println(-1);
                }else{
                    st.pop();
                }
            }else if(caseNu == 3){
                int x = scn.nextInt();
                st.push(x);
            }else{
                if(st.size()==0){
                    System.out.println(-1);
                }else{
                    System.out.println(st.peek());
                }
            }

        }
    }
}
```

Reverse string

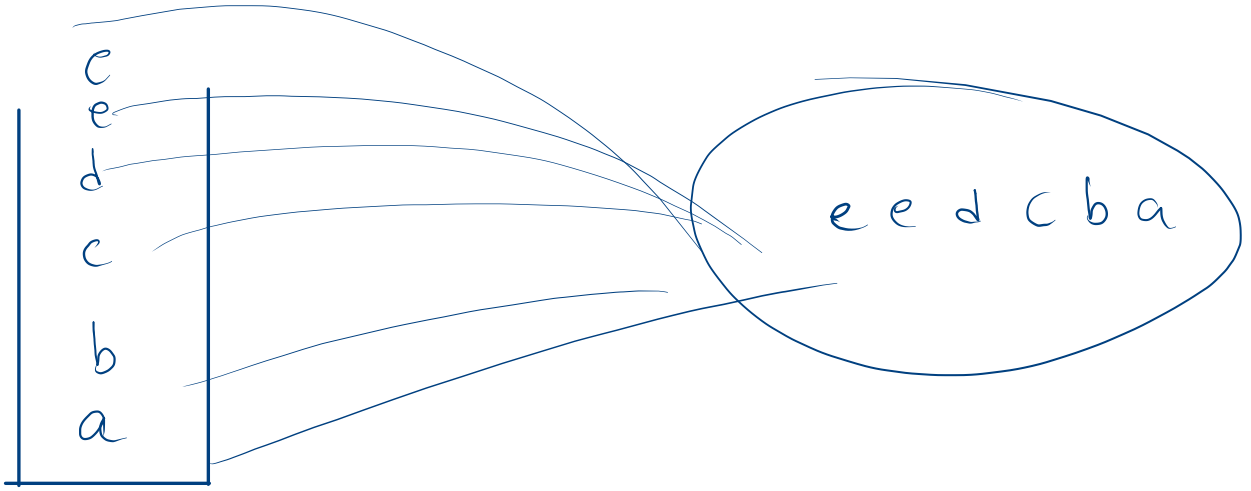
Given a String Str. We have to $Reverse$ the string Str with help of only $stacks$.

Sample Input 0

```
abcdee
```

Sample Output 0

```
eedcba
```

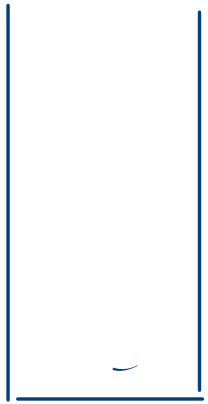


```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        Stack<Character> st = new Stack();
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            st.push(ch);
        }
        String ans = "";
        while(st.size() != 0){
            ans += st.pop();
        }
        System.out.println(ans);
    }
}
```

ans = "edcba"



Delete consecutive

Given a sequence of N $strings$, the task is to check if any two similar words come together then they destroy each other than $print$ the number of words left in the sequence after this $pairwise$ destruction.

Sample Input 0

```
4
aa ab ab ac
```

Sample Output 0

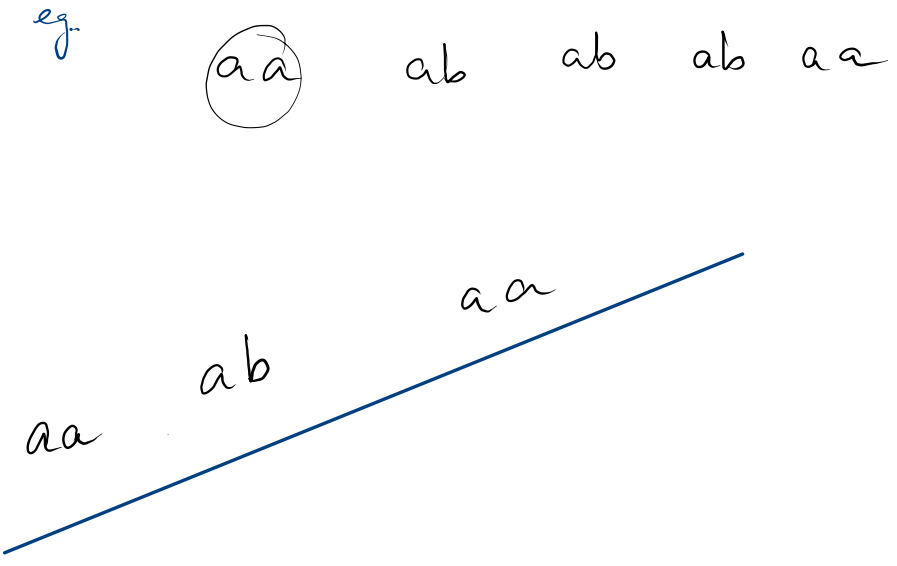
```
2
```

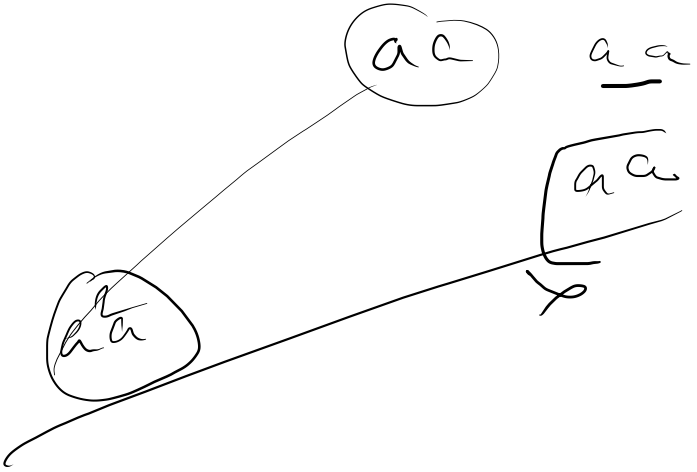
Sample Input 1

```
4
aa ab ab aa
```

Sample Output 1

```
0
```

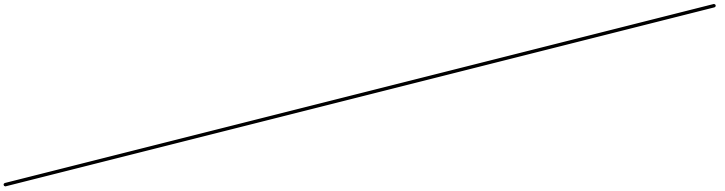




Wrong

aa aa aa

```
8           int n = scn.nextInt();
9           Stack<String> st = new Stack();
10          for(int i = 1; i <= n; i++){
11              String x = scn.next();
12              if(i == 1){
13                  st.push(x);
14              }else{
15                  if(x.equals(st.peek())){
16                      st.pop();
17                  }else{
18                      st.push(x);
19                  }
20              }
21          }
22          System.out.println(st.size());
23      }
24  }
```



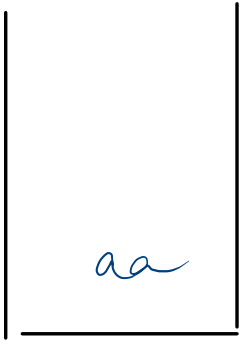
aa aa aa

x = aa

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        Stack<String> st = new Stack();
        for(int i = 1; i <= n; i++){
            String x = scn.next();
            if(i == 1 || st.size() == 0){
                st.push(x);
            }else{
                if(x.equals(st.peek())){
                    st.pop();
                }else{
                    st.push(x);
                }
            }
        }
        System.out.println(st.size());
    }
}
```



Reverse Words in a Given String

eg3 __ am __ cool ___ sunday__

Sample Input 0

```
reverse words in a given string
```

Sample Output 0

```
string given a in words reverse
```

eg1 "am cool sunday"

"sunday cool am"

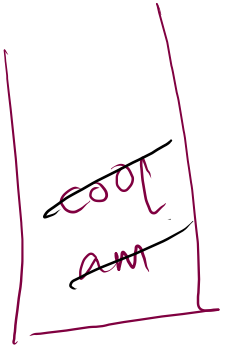
eg2 am __ cool ___ sunday

" am cool sunday "

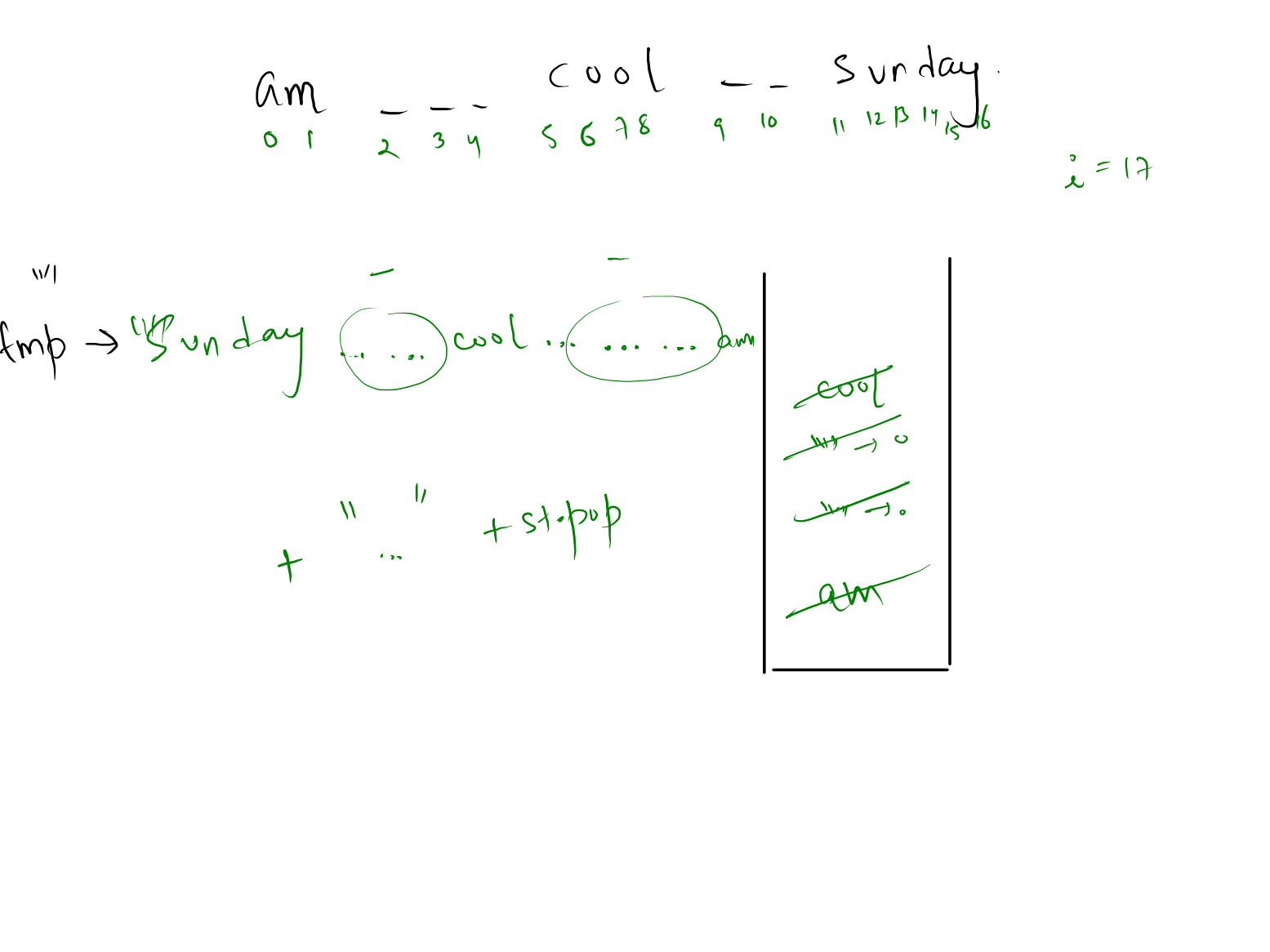
0 1 2 3 4 5 6 7 8 9 10 11 12 13

i = 14

tmp → sunday - cool - am



tmp += " " + st.pop()


am cool Sunday
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
i = 17
tmp → "Sunday cool am
+ " "
+ ...
+ st.pop
cool
am

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.nextLine();
        Stack<String> st = new Stack<>();
        String tmp = "";
        for(int i = 0; i < s.length(); i++){
            if(s.charAt(i) == ' '){
                st.push(tmp);
                tmp = "";
            }else{
                tmp += s.charAt(i);
            }
        }

        while(st.size() != 0){
            tmp += " " + st.pop();
        }
        System.out.println(tmp);

    }
}
```