Revision.

Binary Search.

→ The Banana challenge → koko eating banana

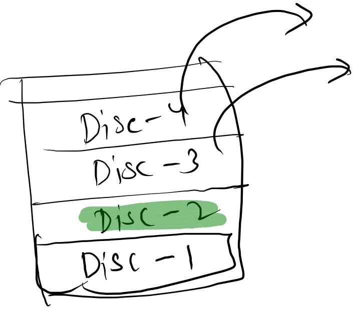
→ Painter Problem.

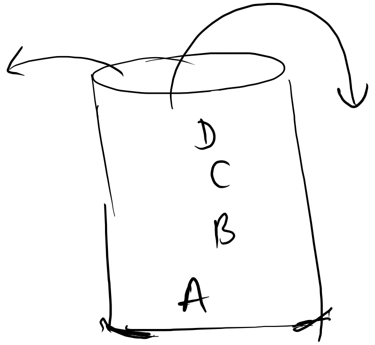
no direct search space!

?

Stack. → Data structure. :→ Dynamic.

↳ Bucket - like DS.

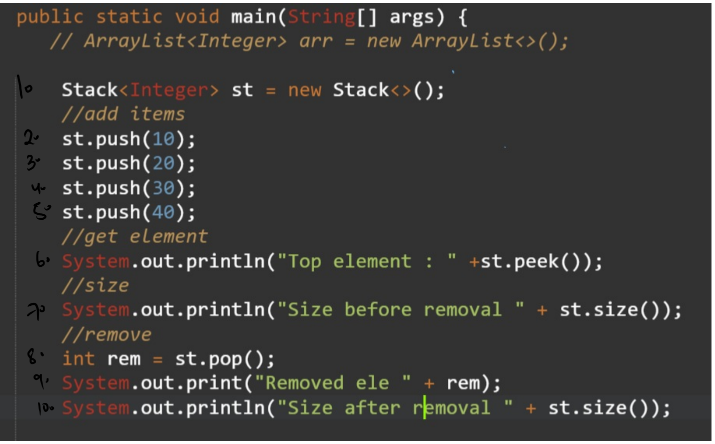




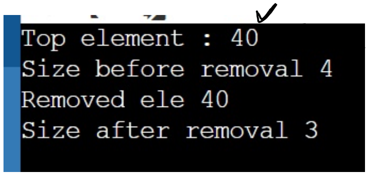
Principle :- LIFO ✓

↳ Last in first out

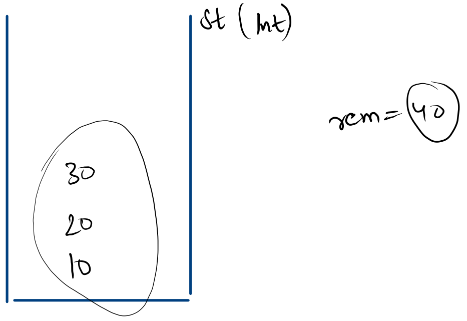
Stack.
- Initialize
- get
- add
- remove
- size



```
public static void main(String[] args) {
    // ArrayList<Integer> arr = new ArrayList<>();

1.    Stack<Integer> st = new Stack<>();
      //add items
2.    st.push(10);
3.    st.push(20);
4.    st.push(30);
5.    st.push(40);
      //get element
6.    System.out.println("Top element : " +st.peek());
      //size
7.    System.out.println("Size before removal " + st.size());
      //remove
8.    int rem = st.pop();
9.    System.out.print("Removed ele " + rem);
10.   System.out.println("Size after removal " + st.size());
```



```
Top element : 40
Size before removal 4
Removed ele 40
Size after removal 3
```



Stack Syntax Learning

Problem   Submissions   Leaderboard   Discussions

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.

boolean

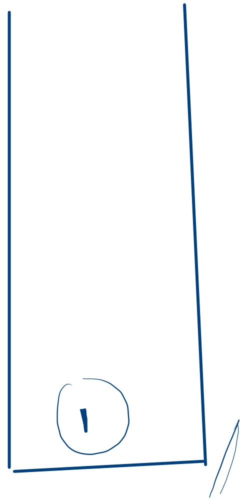
isEmpty() $\to$ T

$\to$ F

T = 10

c , x

3, 1



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

Sample Output 0

```
2
2
1
1
1
```

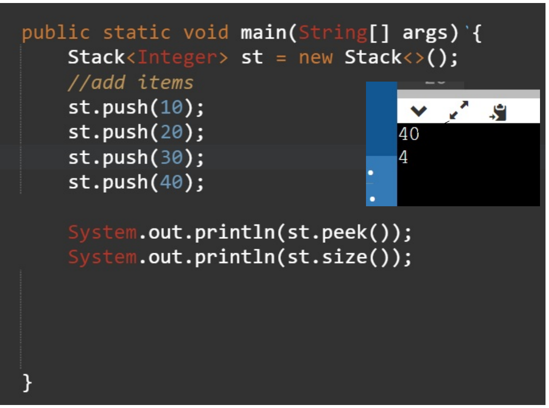
case, x optional

st. peek()

↳ tells us about to element

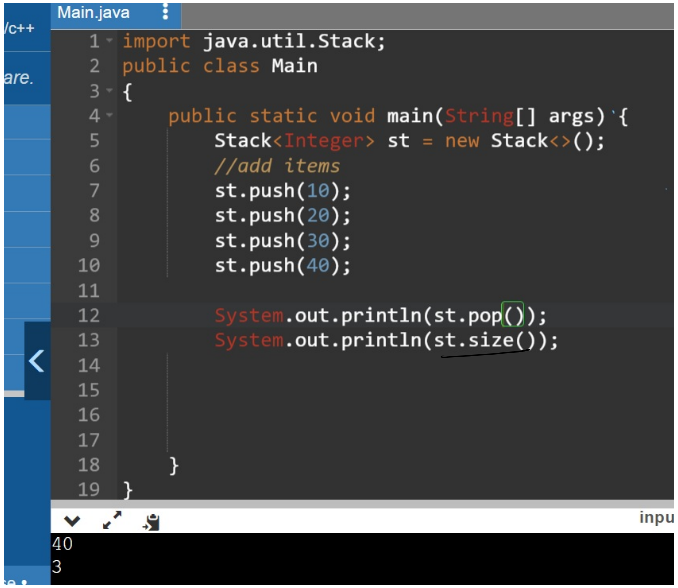
st.pop()

st.peek()
+
remove



```
public static void main(String[] args) {
    Stack<Integer> st = new Stack<>();
    //add items
    st.push(10);
    st.push(20);
    st.push(30);
    st.push(40);

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



}
```

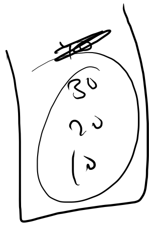
```
40
4
```



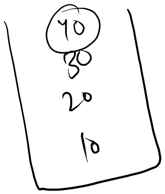
Main.java

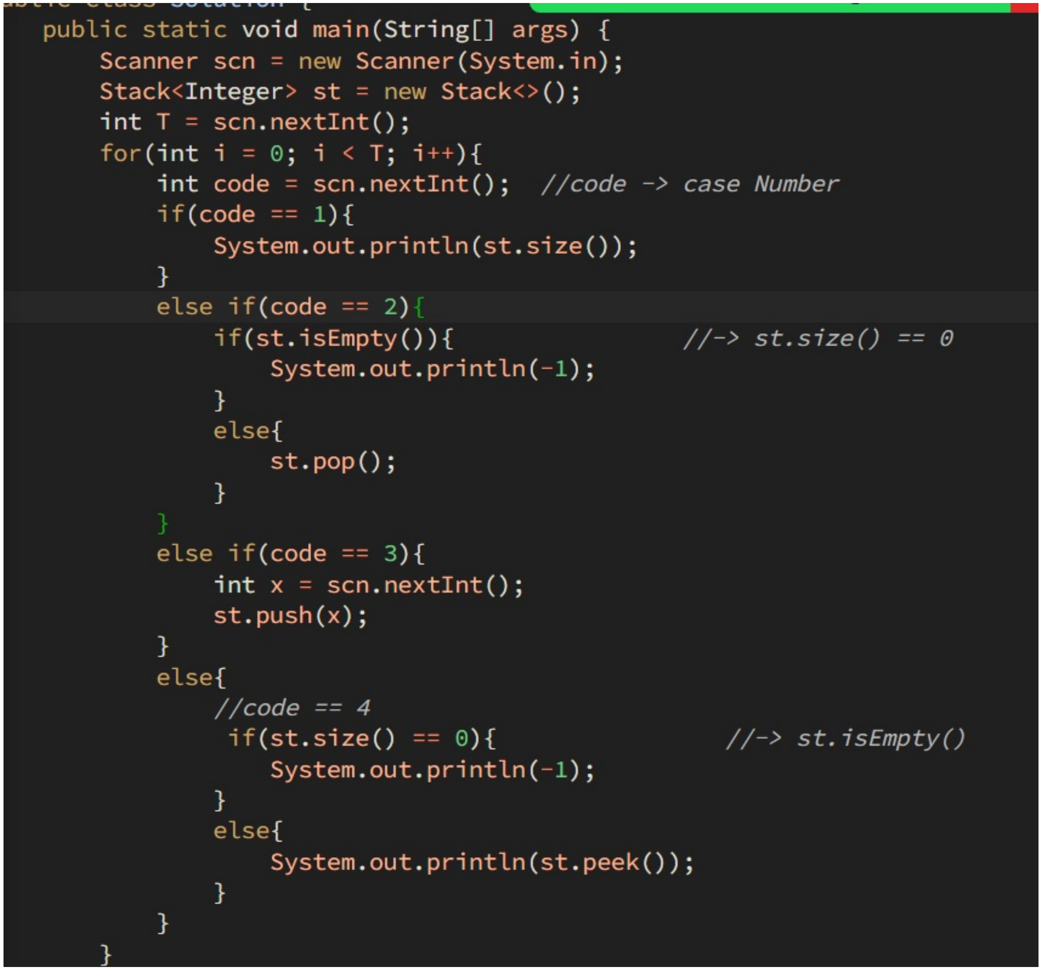
```
import java.util.Stack;
public class Main
{
    public static void main(String[] args) {
        Stack<Integer> st = new Stack<>();
        //add items
        st.push(10);
        st.push(20);
        st.push(30);
        st.push(40);

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





    }
}
```

```
40
3
```



40





```
ublic class Solution {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        Stack<Integer> st = new Stack<>();
        int T = scn.nextInt();
        for(int i = 0; i < T; i++){
            int code = scn.nextInt();  //code -> case Number
            if(code == 1){
                System.out.println(st.size());
            }
            else if(code == 2){
                if(st.isEmpty()){                   //-> st.size() == 0
                    System.out.println(-1);
                }
                else{
                    st.pop();
                }
            }
            else if(code == 3){
                int x = scn.nextInt();
                st.push(x);
            }
            else{
                //code == 4
                 if(st.size() == 0){                  //-> st.isEmpty()
                    System.out.println(-1);
                }
                else{
                    System.out.println(st.peek());
                }
            }
        }
```

Delete consecutive

```
4
aa  ab  ab  ac
```

eg. aa ~~ab ab~~ ~~ac ac~~ aa dc

⇓

~~aa aa~~ dc

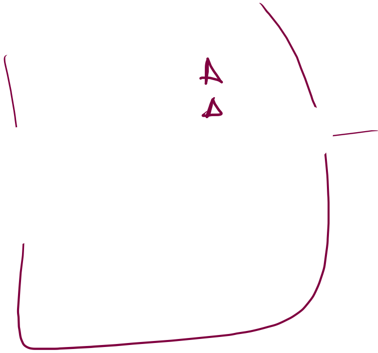
answer = 1

eg2. ~~aa aa~~ aa

answer=1

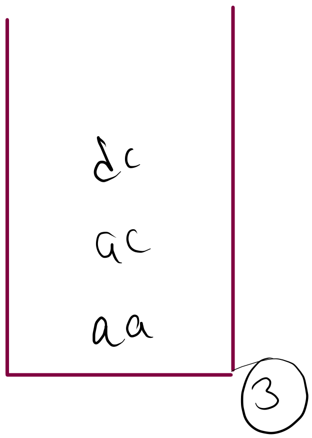
eg3. aa ab aa ab ac dc ac

answer= 7



Delete Consecutive.

$$\begin{array}{cccccc} & & & & & \text{curr} \\ & & & & & \downarrow \\ \text{eg.1} & \underline{\underline{aa}} & ab & ab & ac & dc \end{array}$$

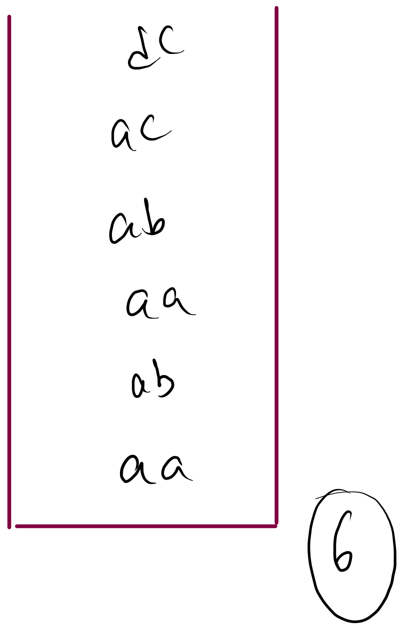


```
if curr.equals(st.peek())
{
    remove
}
else
{
    add
}
```

eg. 2.

aa ab aa ab ac dc

$\uparrow$
c



c == peek
↳ remove

else
↳ add.

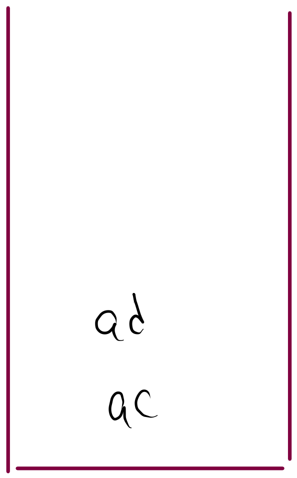
eg. 3.

aa ~~ab~~ ~~ab~~ aa ac ad.

↑
c

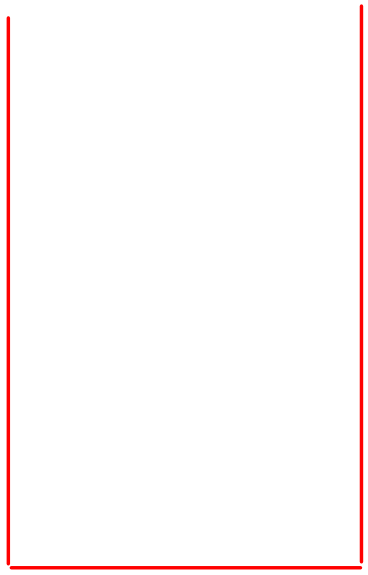
peek == curr
↳ remove

else add



eg. 4.

aa ab ac ac ab aa ↓



Curr == peek

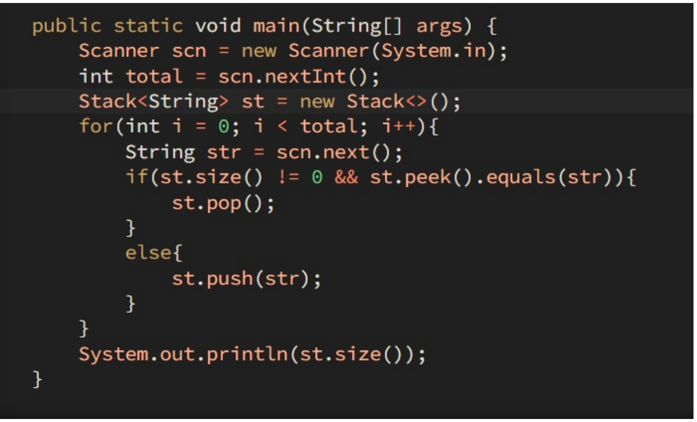
↳ Remove

else add

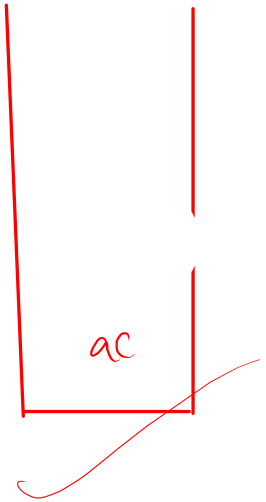
ans = 0

aa   ab   ab   aa   ac

ans = 1



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



Implement a stack using ArrayList

Sample Input 0

```
5
push 1
push 2
display
pop
size
```

add

add

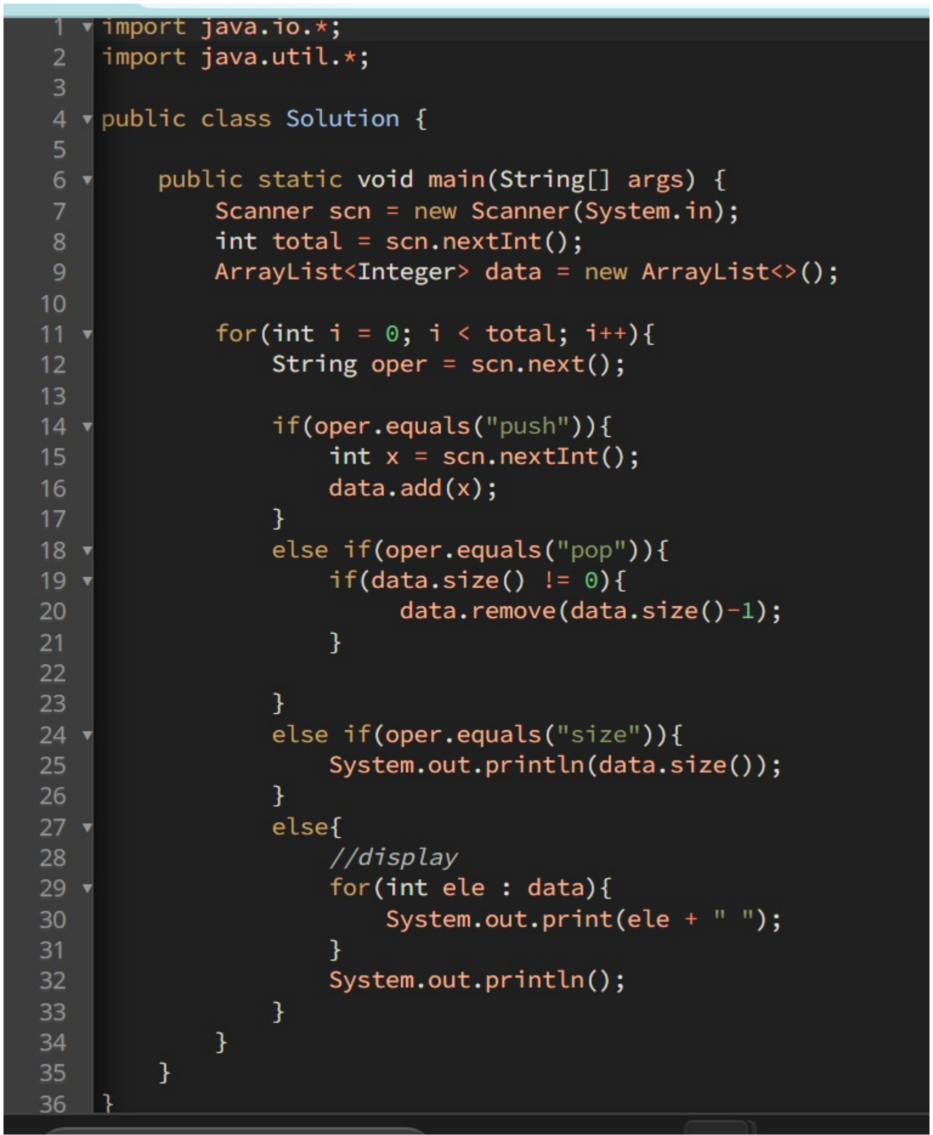
for (each)

remove from behind

size.

Sample Output 0

```
1 2
1
```



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

public class Solution {

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

        for(int i = 0; i < total; i++){
            String oper = scn.next();

            if(oper.equals("push")){
                int x = scn.nextInt();
                data.add(x);
            }
            else if(oper.equals("pop")){
                if(data.size() != 0){
                     data.remove(data.size()-1);
                }

            }
            else if(oper.equals("size")){
                System.out.println(data.size());
            }
            else{
                //display
                for(int ele : data){
                    System.out.print(ele + " ");
                }
                System.out.println();
            }
        }
    }
}
```