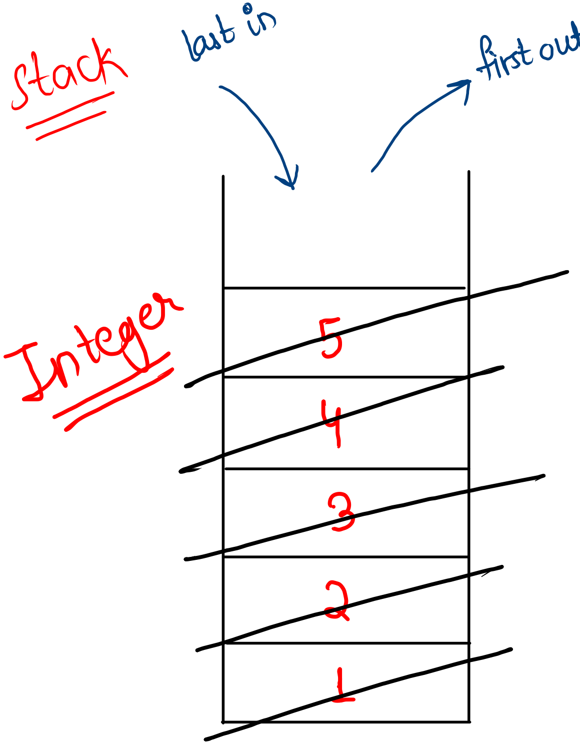
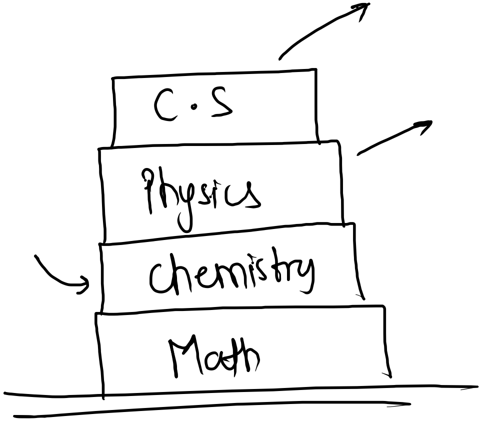
⇒ Stack (LIFO) or (FILO)

↳ last in, first out ↳ First in, last out



(chapati box)



// extract an element

always from top

5 4 3 2 1

⇒ Stack (also dynamic in nature)

↳ It's a data Structure similar to arraylist

↳ only difference is, we can put or remove elements from 1 side of stack

↳ It follows LIFO order

⇒ Syntex and Inbuilt functions

```
Stack< Object> st = new Stack<>();
```

↳ Integer
↳ Boolean
↳ String etc.

advantage :-

$T.C = O(1)$

To add element in Stack : `st.push(value);`

To remove element from stack : `st.pop(); //returns the element`

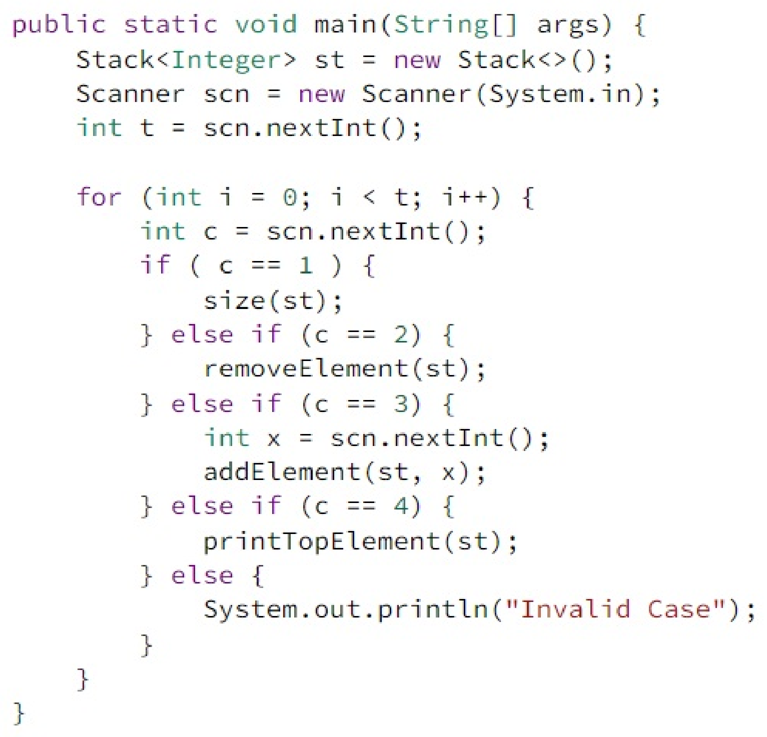
To get top element from stack : `st.peek(); //returns top element`

To get the size of stack : `st.size(); //return size of st`

To check if stack is empty or not : `st.isEmpty(); //return true if st is empty & false otherwise`

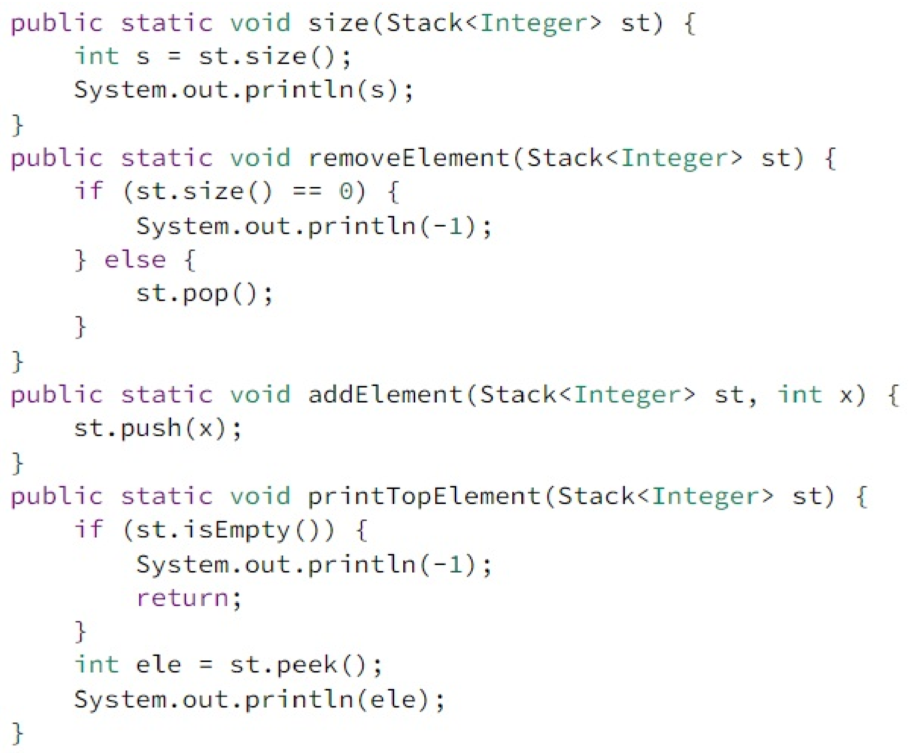
Stack Syntax Learning

code



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

    for (int i = 0; i < t; i++) {
        int c = scn.nextInt();
        if ( c == 1 ) {
            size(st);
        } else if (c == 2) {
            removeElement(st);
        } else if (c == 3) {
            int x = scn.nextInt();
            addElement(st, x);
        } else if (c == 4) {
            printTopElement(st);
        } else {
            System.out.println("Invalid Case");
        }
    }
}
```



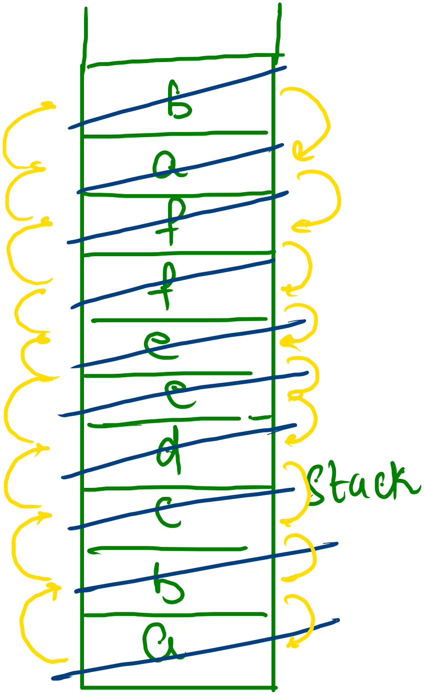
```java
public static void size(Stack<Integer> st) {
    int s = st.size();
    System.out.println(s);
}
public static void removeElement(Stack<Integer> st) {
    if (st.size() == 0) {
        System.out.println(-1);
    } else {
        st.pop();
    }
}
public static void addElement(Stack<Integer> st, int x) {
    st.push(x);
}
public static void printTopElement(Stack<Integer> st) {
    if (st.isEmpty()) {
        System.out.println(-1);
        return;
    }
    int ele = st.peek();
    System.out.println(ele);
}
```

Reverse string

str = "abcdeeffab";

ans = "baffeedcba";

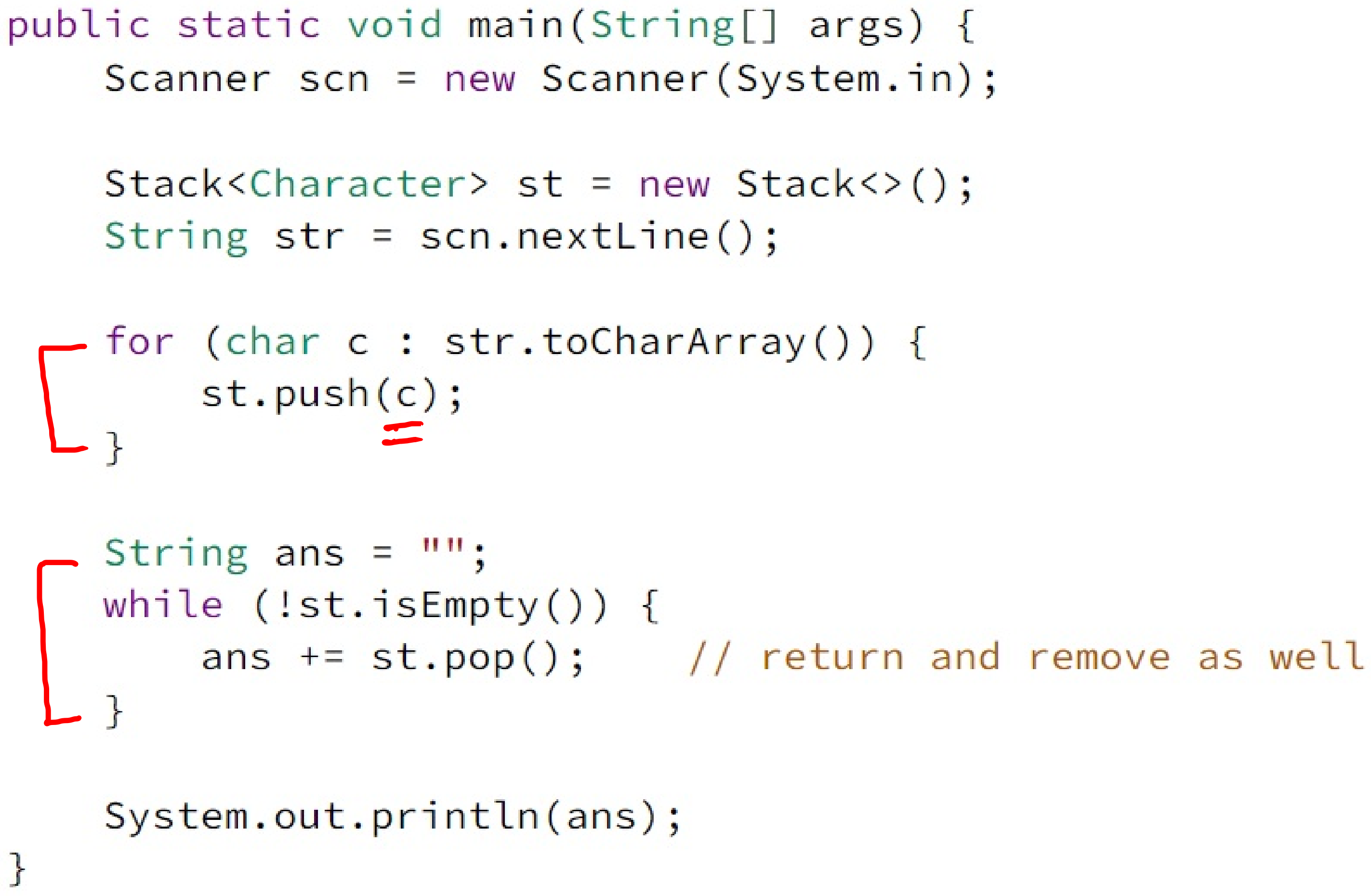
Stack<Character> st = new Stack<>();



ans = baffeedcba

Code

```
for(int i=0; i< str.length(); i++){
    st.push(str.charAt(i));
}
```



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    Stack<Character> st = new Stack<>();
    String str = scn.nextLine();

    for (char c : str.toCharArray()) {
        st.push(c);
    }

    String ans = "";
    while (!st.isEmpty()) {
        ans += st.pop();    // return and remove as well
    }

    System.out.println(ans);
}
```

Delete consecutive

abc ~~efg~~ ~~efg~~ xyz

Ex1

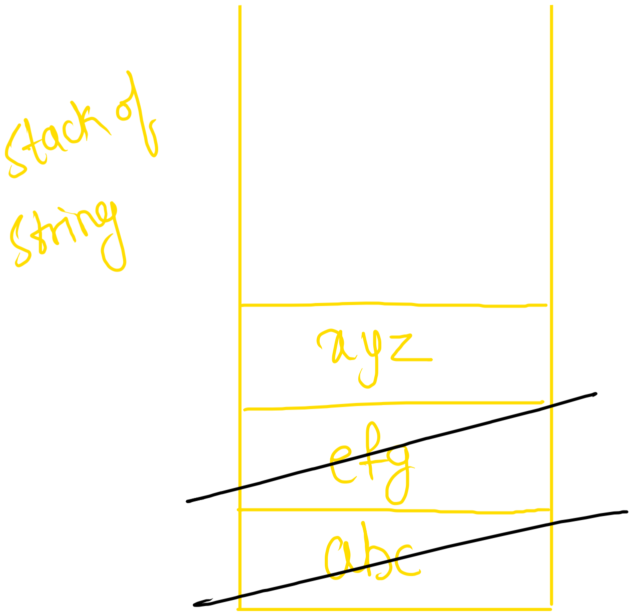
abc xyz → 2

~~abc~~ ~~efg~~ ~~efg~~ ~~abc~~ xyz

Ex2

xyz → 1

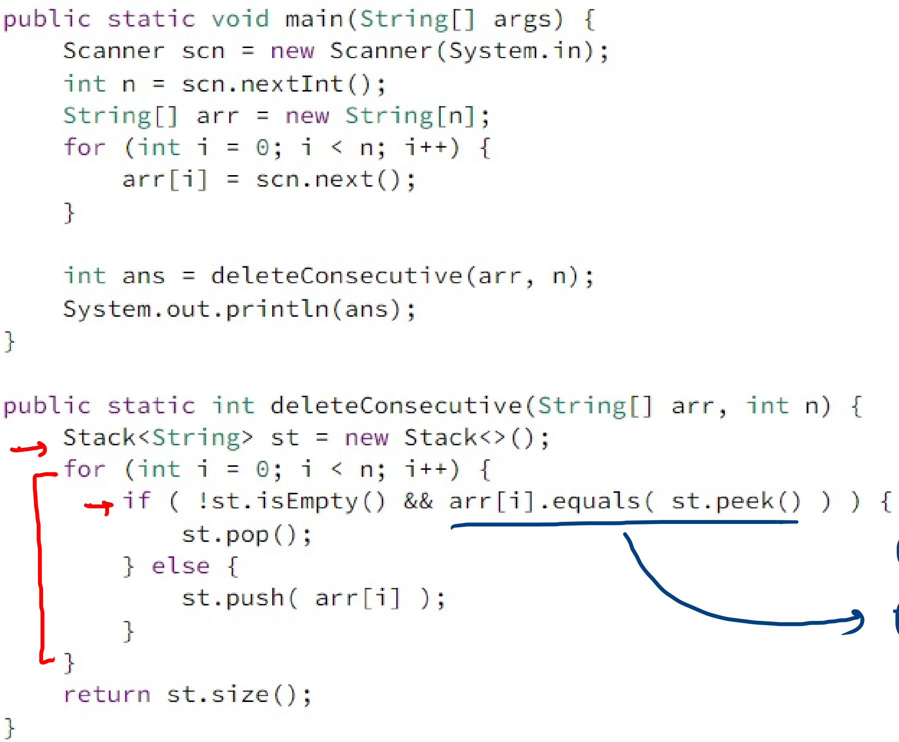
arr	abc	efg	efg	abc	xyz
	0	1	2	3	4



return size of stack

psudo code

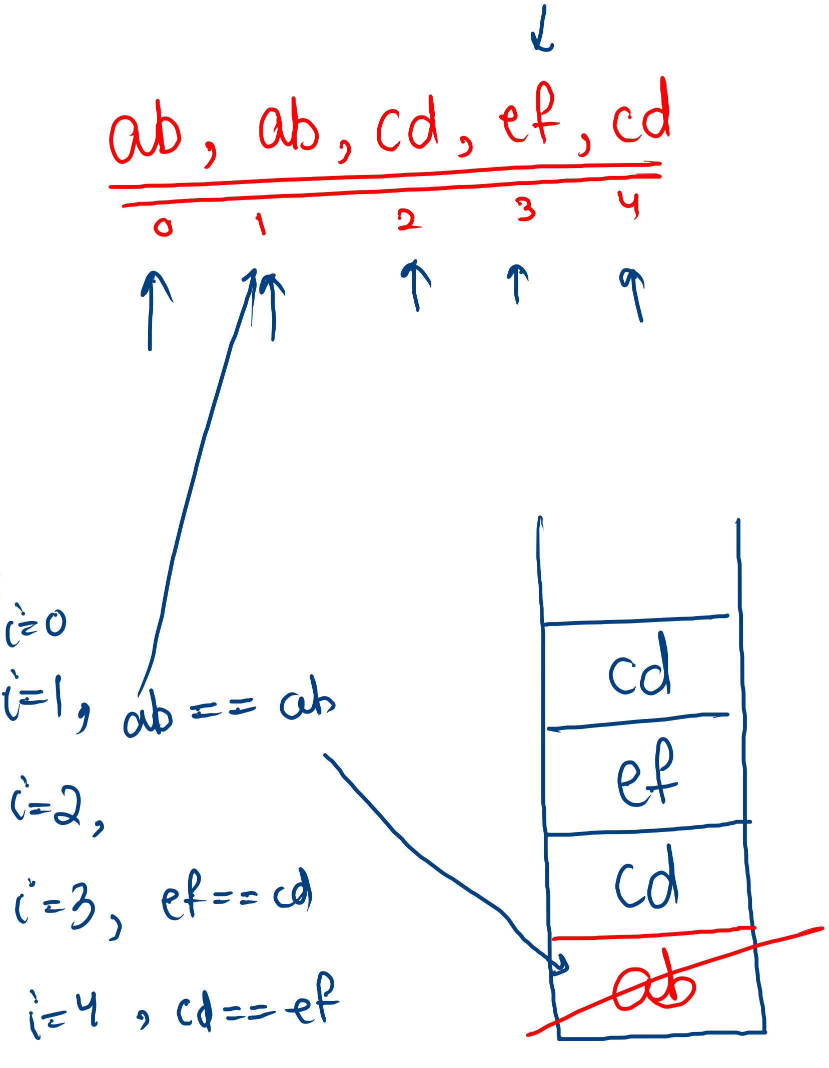
```
1) traverse is array
    1.1) curr == top
        1.1.1) remove top element
    1.2)
        push element
```



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    String[] arr = new String[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.next();
    }

    int ans = deleteConsecutive(arr, n);
    System.out.println(ans);
}

public static int deleteConsecutive(String[] arr, int n) {
    Stack<String> st = new Stack<>();
    for (int i = 0; i < n; i++) {
        if ( !st.isEmpty() && arr[i].equals( st.peek() ) ) {
            st.pop();
        } else {
            st.push( arr[i] );
        }
    }
    return st.size();
}
```



ans = 3