Stack LIFO Data Structure

Last In First Out

Uses: 1) Recursion Call Stack

2) Undo, Redo

3) Browser History.

Operations:

push(x)

pop()

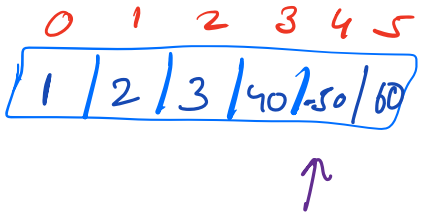
top()

isEmpty()

size()

Implementations

- Stack Using Array



Walmart
VMware
SAP Labs
MMT

```
Class stack {
int size → fixed size
int arr[size]
int top = -1 → the array index where my top elem is
```

1, 2, 3

1 2 3

```
void push(int x) {
if( top == size-1)
    sout("Stack full") } stack overflow
else
    top++      arr[top] = x
}
void pop() {
  if(top == -1) sout("Empty stack") } stack underflow
  else top--
}
int top() {
if(top == -1) sout("Empty stack")
else return arr[top]
}
```

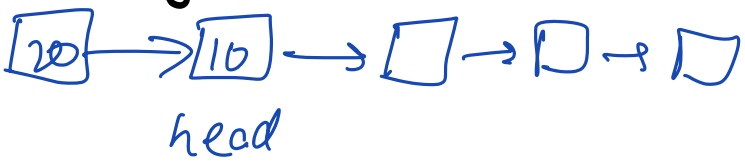
Note: You can use inbuilt stack from Java Collections

Stack with Linked List

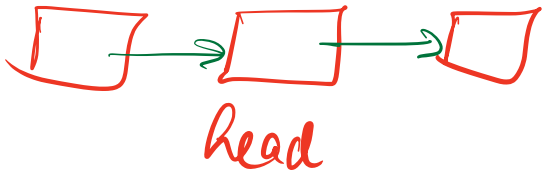
```
class Stack {
    int size = 0
    Node head = null
void push (int x) {
    Node newhead = new Node(x)
    newhead.next = head
    head = newhead
    size++
}
```

```
class Node {
    int data
    Node next

    Node (int x) {
        this.data = x
        this.next = null
    }
}
```



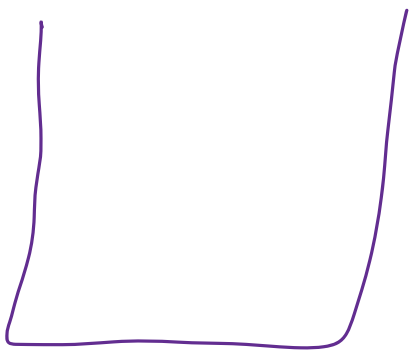

```
void pop () {
    if (size == 0) sout ("Empty stack")
    else {
        head = head.next
        size--
    }
}
```




```
int top () {
    if (size == 0) sout ("Empty stack")
else return head.data
}
```

(x

(x y)



Q Balanced paranthesis

Check if paranthesis is balanced or not

Eg {()[]} ans ⇒ true

{[][)} ans ⇒ false

Soln ⇒ open bracket ⇒ push in stack

closed bracket ⇒ S.top() should be matching open bracket

at the end, if stack empty ⇒ true

Code

```
stack<char> st
for(i=0;i<n;i++){
    if(s[i] is opening bracket)
        st.push(s[i])
    else{   // s[i] is a closing bracket
        if(st.top() is matching open bracket)
            st.pop()
        else
            return false
    }
}
if(st.empty()) return true
else return false
```

TC: O(n) SC: O(n)

Q. Remove equal pair of consecutives

Eg a b c d d c → a b c c

→ a b

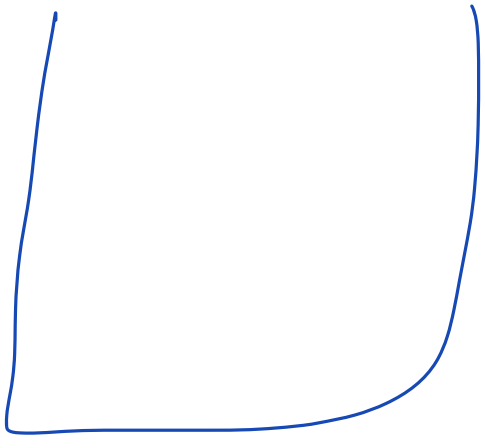
c x

Idea ⇒ stack.

If s[i] == st.top() pop()

else insert s[i] into stack

a b b c b b c a c x



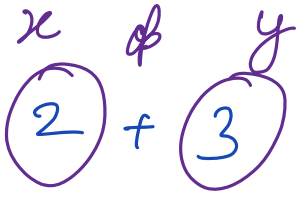
ans = x C

return reverse(ans)

Code

```
stack<char> st
for(i=0;i<n;i++){
    if(!st.empty && st.top()==s[i])
        st.pop()
    else
        st.push(s[i])
}
string ans=""
while(!st.empty()){
    ans+= st.top()
    st.pop()
}
return reverse(ans)
```

TC, SC: O(n)

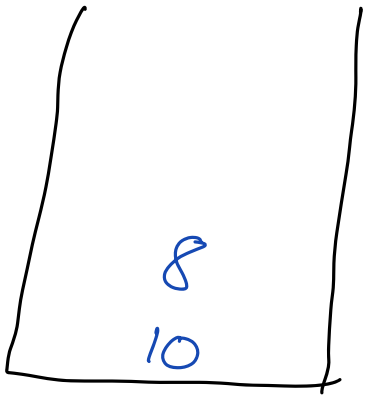


(infix) x op y

(postfix) x y op

Q Post fix eval

3, 7, +, 2, 4, *, -

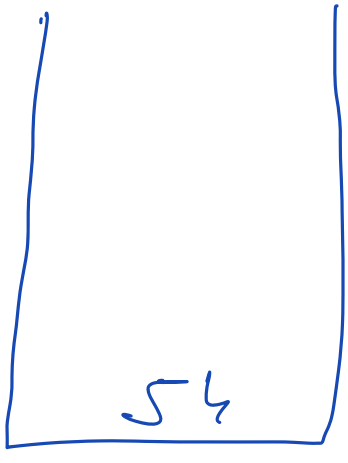


- If number, put in stack
- If operator get top 2 from stack
  do ⇒ second top op top
  insert back result

Eg ⇒ 3 7 5 8 4 / + * 2 + +

Code

```
for (i=0; i<n; i++) {
  if (s[i] is num) {
    st.push (s[i])
  else {
    y = st.top()
    st.pop()
```



```
x = st.top()
st.pop()
z = x op y
st.push(z)
}
}
return st.top()
```

if-else

TC: $O(n)$
SC:

{done}

3 5 + 2 − 2 5 * −