

09/03/2024

Page No.	
Date	

Day 5 OF DSA

Check
Box

Stack



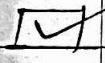
Basic stack operations, Push, pop



Stack Implementation, Using arrays,
linked list.



Implement push, pop operations using
arrays.

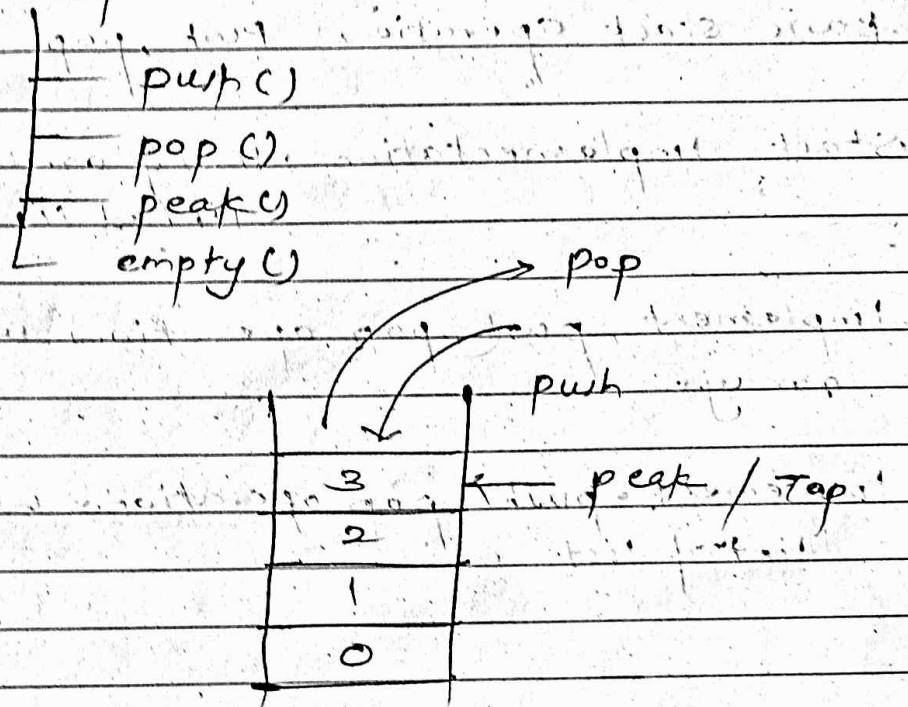


Implement push, pop operations using
linked list.

* stack - LIFO order

LIFO order Data structure

* Operation:



* STL:-

```
stack<int> s;
```

```
s.push(10);
```

```
s.push(20);
```

```
s.pop();
```

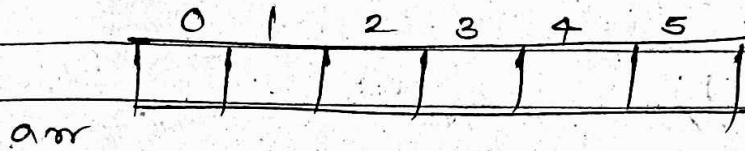
```
s.top;
```

```
if(s.empty()) { cout<<"Empty"<<endl; }
else { cout<<"Not Empty"<<endl; }
```

* Implementation of stack without STL

* Arrays

* Linked list

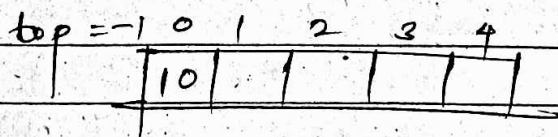


top = -1 → Index

```
int top;
int arr[5];
int size;
```

* Push :-

- ① check space is available or not.
- ② top++
- ③ arr[top] = 22;



top++ → top = 0
arr[top] = 10

$(top \geq 0)$

* Pop();



① Check element is present or not

② $top--$;

* empty();

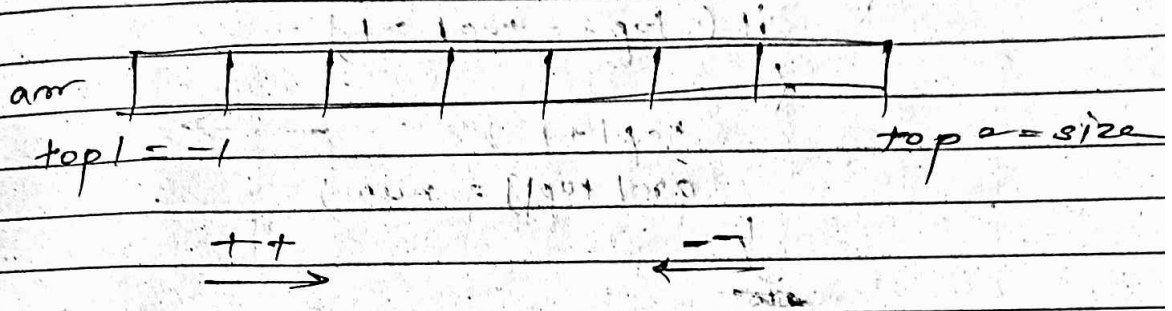
$top = -1$ ✓

* peak();

$arr[top]$ ✓

Time Complexity : $O(1)$

* Two stacks (One top) (One bottom)



class TwoStack {

```

    public TwoStack (int arr, int top1, int top2, int size) {
        // Initializat
        this.arr = arr;
        this.top1 = top1;
        this.top2 = top2;
        this.size = size;
    }

```

public {

TwoStack (int s)

```

    {
        this.size = s;
        top1 = -1; // Constructor
        top2 = s;
        arr = new int [s];
    }

```

$$arr[top] = num;$$

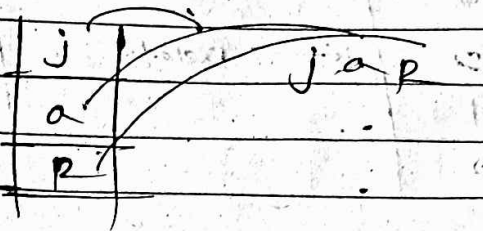
$arr[top2] = num;$

if return ans; } else return -1;

↳ ease of return; y

* Reverse a string using stack,

$raj = jar$

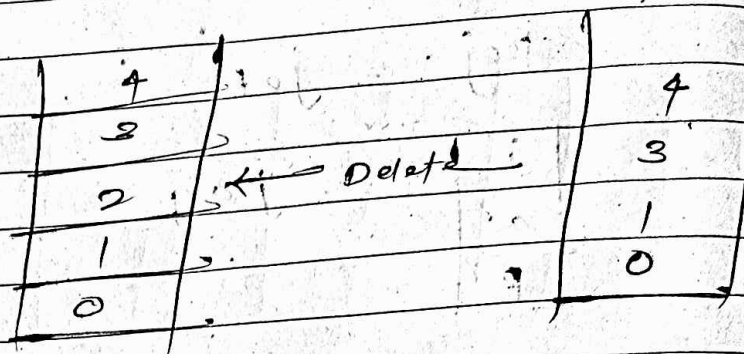


Approach 1

- ① put all characters of string in stack
- ② Take new string $s = ""$
- ③ In that put all top element & pop again & again.
- ④ & print that string;

Time complexity : $O(N)$ space complexity : $O(N)$

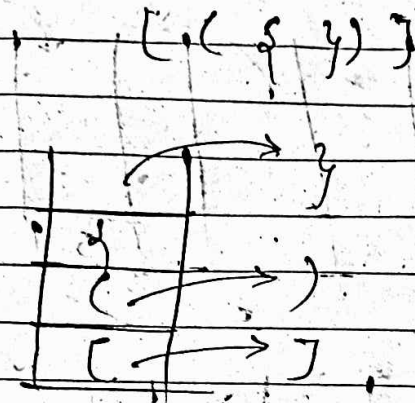
* Delete middle element from stack :-



Approach 1

- ① Remove top element & store it in new variable
- ② after & before it check conditⁿ
 if $\text{count} = \text{size} / 2$
 then pop that element
- ③ After that recursively call that functⁿ
- ④ & after ending of it add the 1st element that we removed, in stack.

* Valid Parenthesis

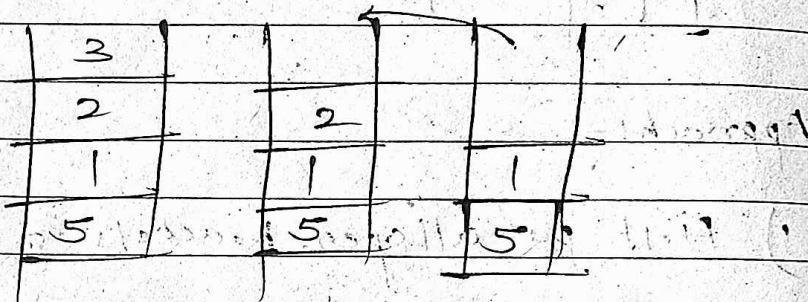
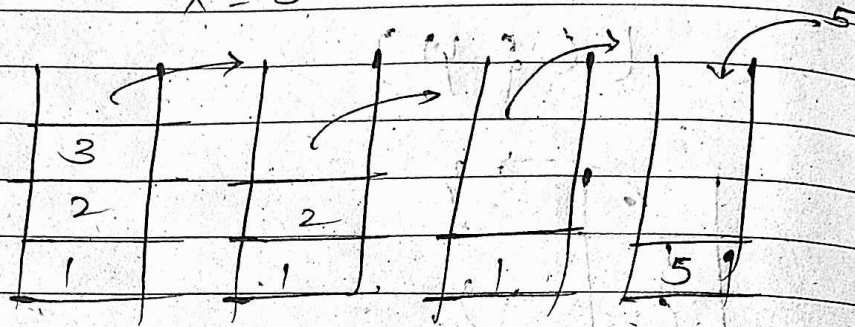


Approach:-

- 1) First put all open brackets in stack
- 2) Then pop that bracket using closing brackets
- 3) After check stack is empty or not.
If it is empty then it is valid parenthesis

* Insert element at Bottom

$$X = 5$$



Approach:

- 1) pop all elements from stack till it is not empty will
- 2) then push our X.
- 3) After that whatever we pop in that way we push this elements.