09/08/2024

Page No.
Date

Day 12 OF DSA.

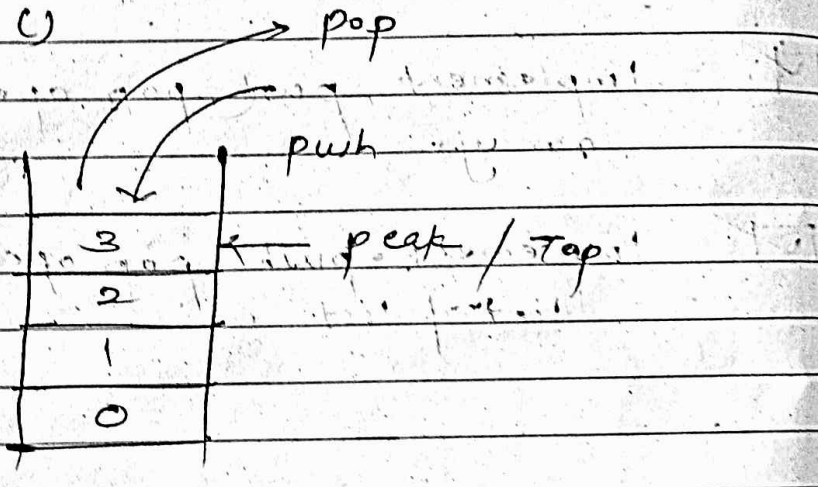
Check Box	stack
[✓]	Basic stack operations, Push, pop
[✓]	Stack Implementation, Using arrays, linked list.
[✓]	Implement push, pop operations using arrays.
[✓]	Implement push, pop operations using linked list.

* stack :-

LIFO order Data structure.

* Operations

- push()
- pop()
- peak()
- empty()



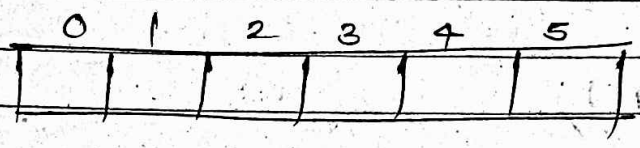
* STL :-

```
stack<int> s;
s.push(10);
s.push(20);

s.pop();

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

Page No.
Date

* Implementation of stack without STL

* Arrays
* Linked list.

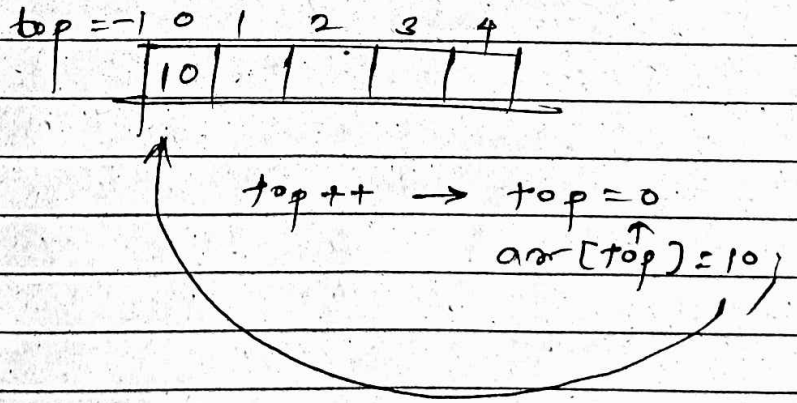


top = -1 → Index

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

* push :-

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



(top >= 0)

* pop();

① check element is present or not

② top--;

* empty();

top = -1

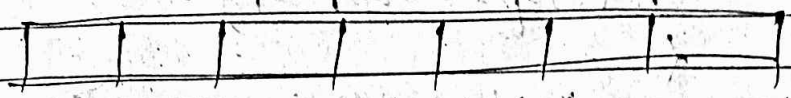
* peak();

arr[top];

Time complexity : O(1)

Page No.
Date

Two stacks



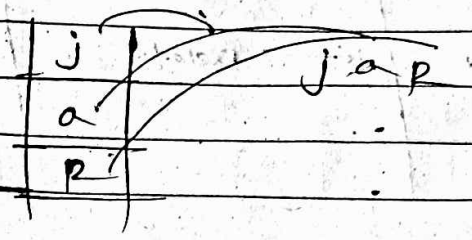

```
class twostack
{
  public
    int *arr;
    int top1;        // Initialization
    int top2;
    int size;

  public:
    TwoStack(int s)
    {
      this->size = s;
      top1 = -1;     // Constructor
      top2 = s;
      arr = new int[s];
    }
```

```
void push1(int num)
{
    if(top2 - top1 > 1)
    {
        top1++;
        arr[top1] = num;
    }
}
void push2(int num)
{
    if(top2 - top1 > 1)
    {
        top2--;
        arr[top2] = num;
    }
}
int pop1()
{
    if(top1 >= 0)
    {
        int ans = arr[top1];
        top1--;
        return ans;
    }
    else return -1;
}
int pop2()
{
    if(top2 < size)
    {
        int ans = arr[top2];
        top2++;
        return ans;
    }
    else return -1;
}
```

Page No.
Date

* Reverse a string using stack,

raj = jar



Approach:

(1) put all characters of string in stack.

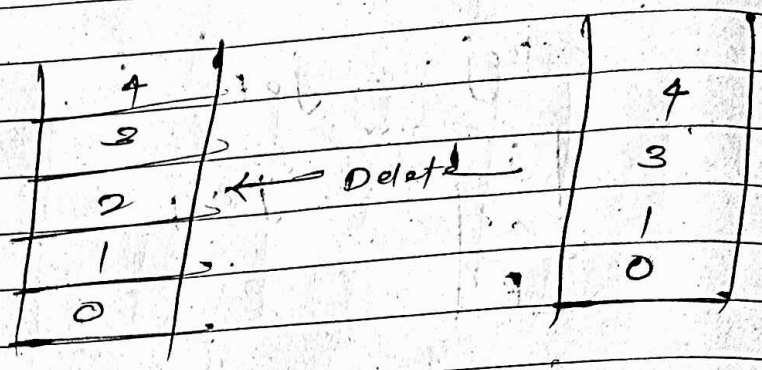
(2) Take new string = ""

(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) Remove top element & store it in new variable

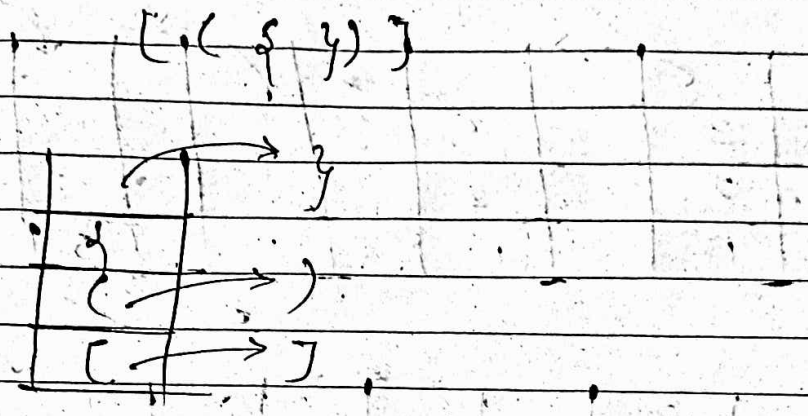
(2) ~~After~~ & before it check condition if count == size/2 then pop that element.

(3) After that recursively call that funct^n.

(4) & after ending of it add the popped element that we removed, in stack.

Page No.
Date

* Valid Parenthesis



Approach :-

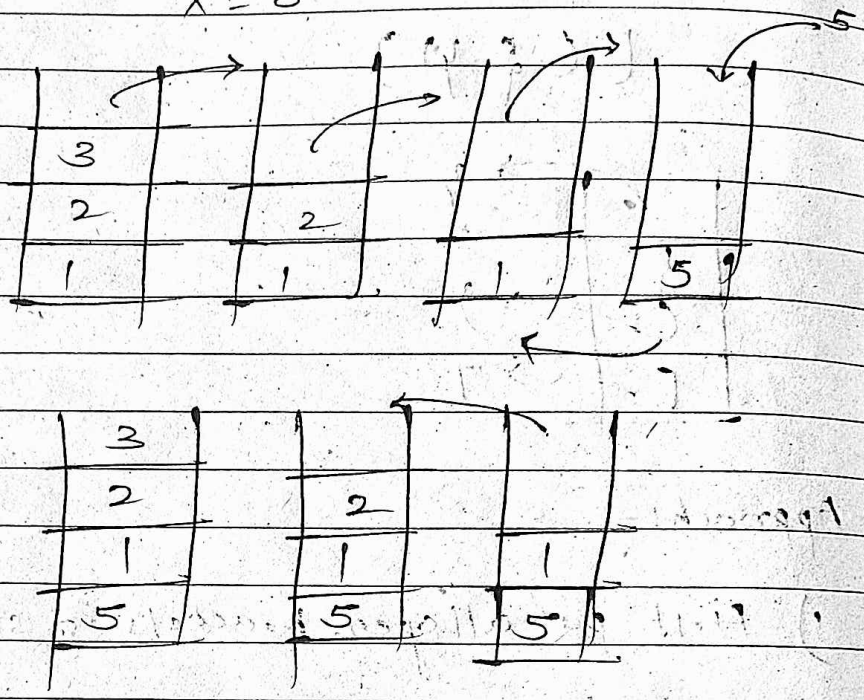
1) First put all open brackets in stack

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.