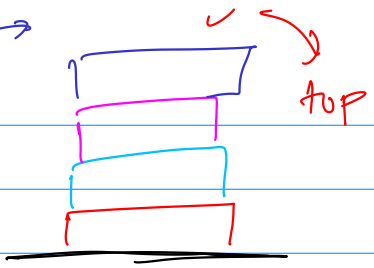
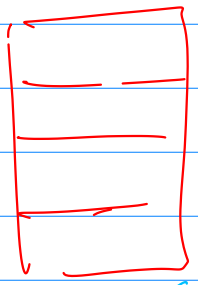
Stack Data Structure



insert push
remove pop

push(5)
push(2)
top



push(c)
push(3)

push(1) → stack overflow.

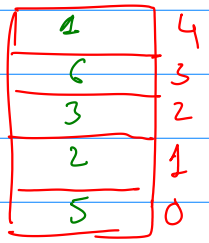
pop() → 3
pop() → 6

pop() → 2
pop() → 5

pop()
→ stack underflow.

size = 5
top = 4
top == (size - 1)

full

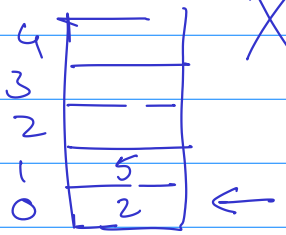


(index) top = 4 ← empty

5

int *arr = new int[size];

size = 5



worst case

push(5)
push(2)
push(3)

extra operations
slow

pop() → 3

```
Class Stack
{ private:
    int *arr;
    int size;
    int top;
  public:
    Stack(int s)
    {
        size = s;
        top = -1;
        arr = new int[size];
    }
    boolean IsEmpty( ); ← do it
    boolean IsFull( ); ← do it
    :
    :
}
```

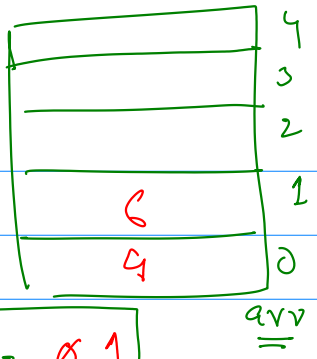
size = 5
top = 4
if top == (size - 1)
return true

full ?

push(4)

push(6)

```
void push ( int val)
{
  if ( IsFull ( ))
  {
    cout<<"Stack overflow "<<endl;
    return;
  }
  top++;
  arr[top] = val;
}
```



top = -1

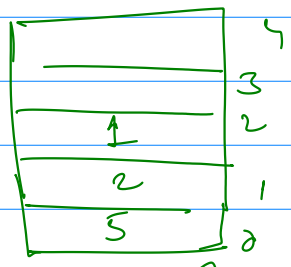
arr[++top] = val; ✓

arr[top++] = val; ✗

top == -1 ?

```
int pop ( )
{
  if ( IsEmpty ( ))
  {
    cout<<"Stack Underflow"<<endl;
    return;
  }
  return arr[top--];
  // int temp = arr[top];
  // top--;
  // return temp;
}
```

top = -1 ✗ -1



Top()

remove ?

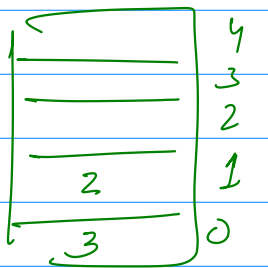
[return arr[top]
top--]

pop() → 1
pop() → 2
pop() → 5
pop() → stack underflow

→ size = 5

```
};
int main ( )
{
  Stack S1(5);
  S1.pop();
  S1.push(3);
  S1.push(2);
  cout<< S1.pop(); → 2
  Stack S2(2);
  S2.push(100);
}
```

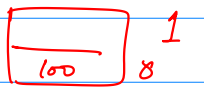
underflow



top = -1 ✗ 0

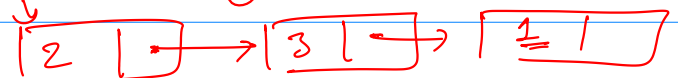
S1

top = 0



insert(5 (val), 1 (pos)) 2 if(pos==1)

Stack using Linked list



top

Push(2)
Push(3)
Push(1)
Pop()

Overflow X

Size

Top()

Pop 2

Pop

Underflow

IsEmpty()

```
node * curr = head;
head = head -> next;
int val = curr -> data;
delete curr;
return val;
```