

# STACK

---

## Problem Solving with Computers-II

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

## Results for **Santa Barbara, CA**

11 PM

2 AM

5 AM

8 AM

11 AM

2 PM

5 PM

8 PM

Sun



59° 55°

Mon



59° 51°

Tue



58° 45°

Wed



59° 45°

Thu



62° 44°

Fri



61° 42°

Sat



63° 42°

Sun



65° 43°

<https://leetcode.com/problems/daily-temperatures/>



A stack is like a chisel —  
a simple tool but you can create/solve complex things!

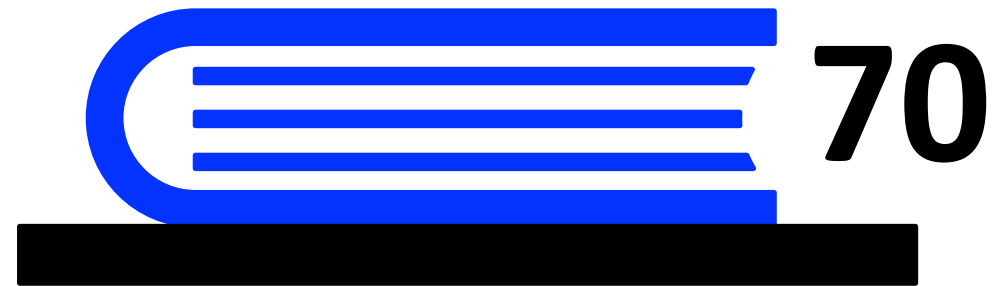
```
stack<int> s
```

**Empty stack**



Operations: **push()**    **pop()**    **top()**

```
stack<int> s  
s.push(70)
```



Operations: **push()**    pop()    top()

```
stack<int> s
```

```
s.push(70)
```

```
s.push(50)
```



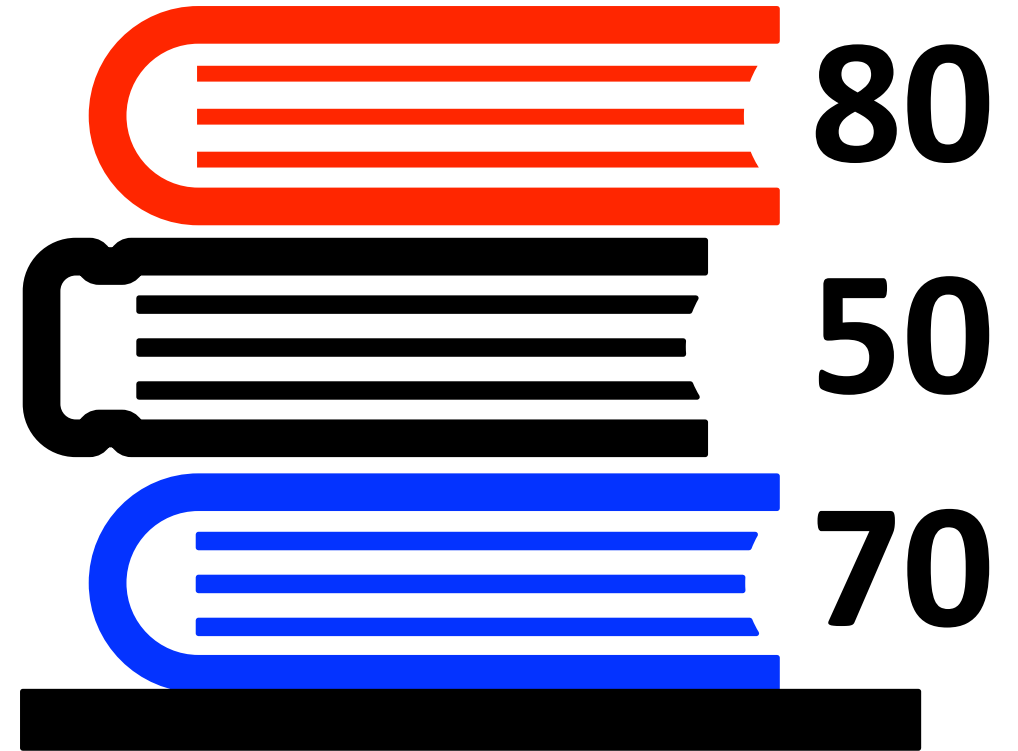
Operations: **push()**    **pop()**    **top()**

```
stack<int> s
```

```
s.push(70)
```

```
s.push(50)
```

```
s.push(80)
```



Operations: **push()**    **pop()**    **top()**

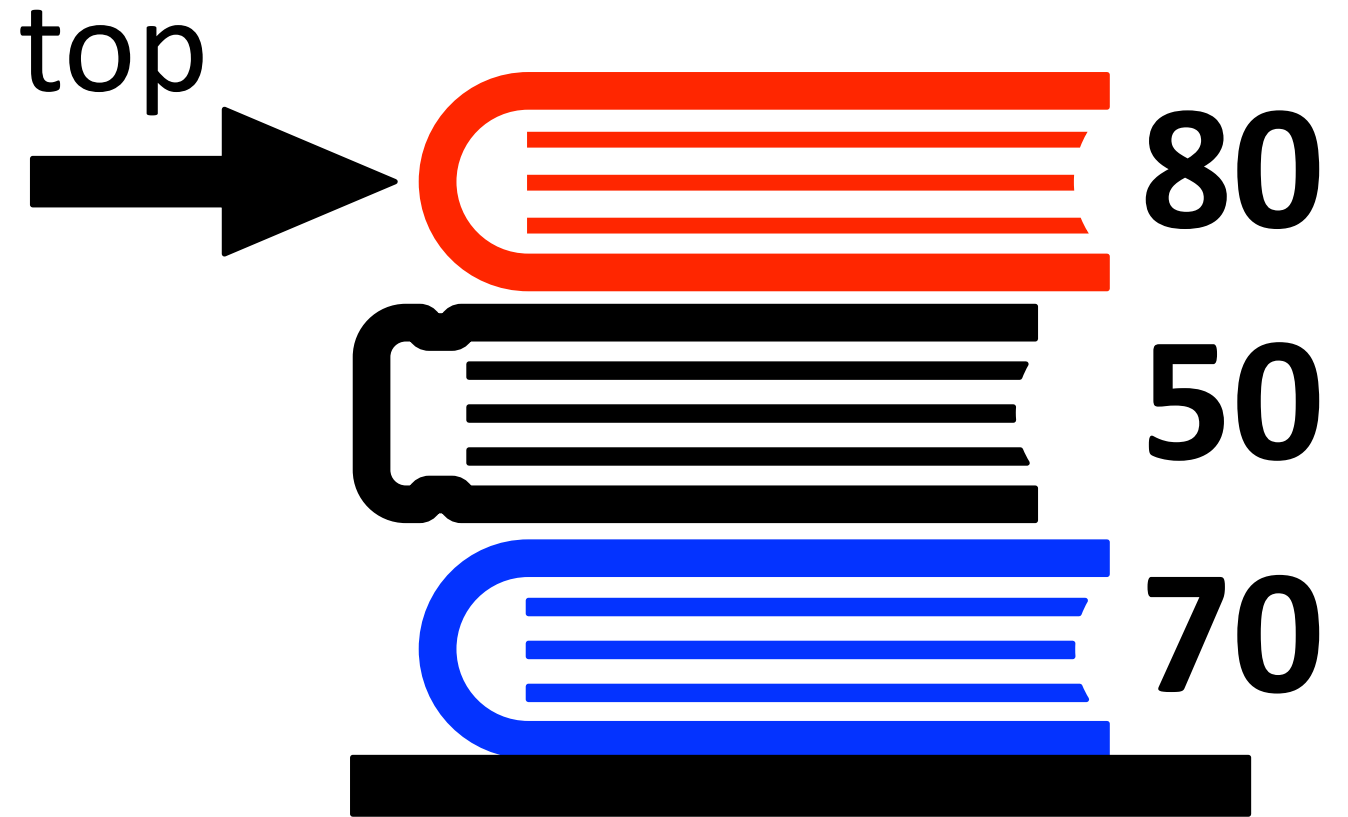
```
stack<int> s
```

```
s.push(70)
```

```
s.push(50)
```

```
s.push(80)
```

**s.top() returns 80**



Operations: **push()**    **pop()**    **top()**



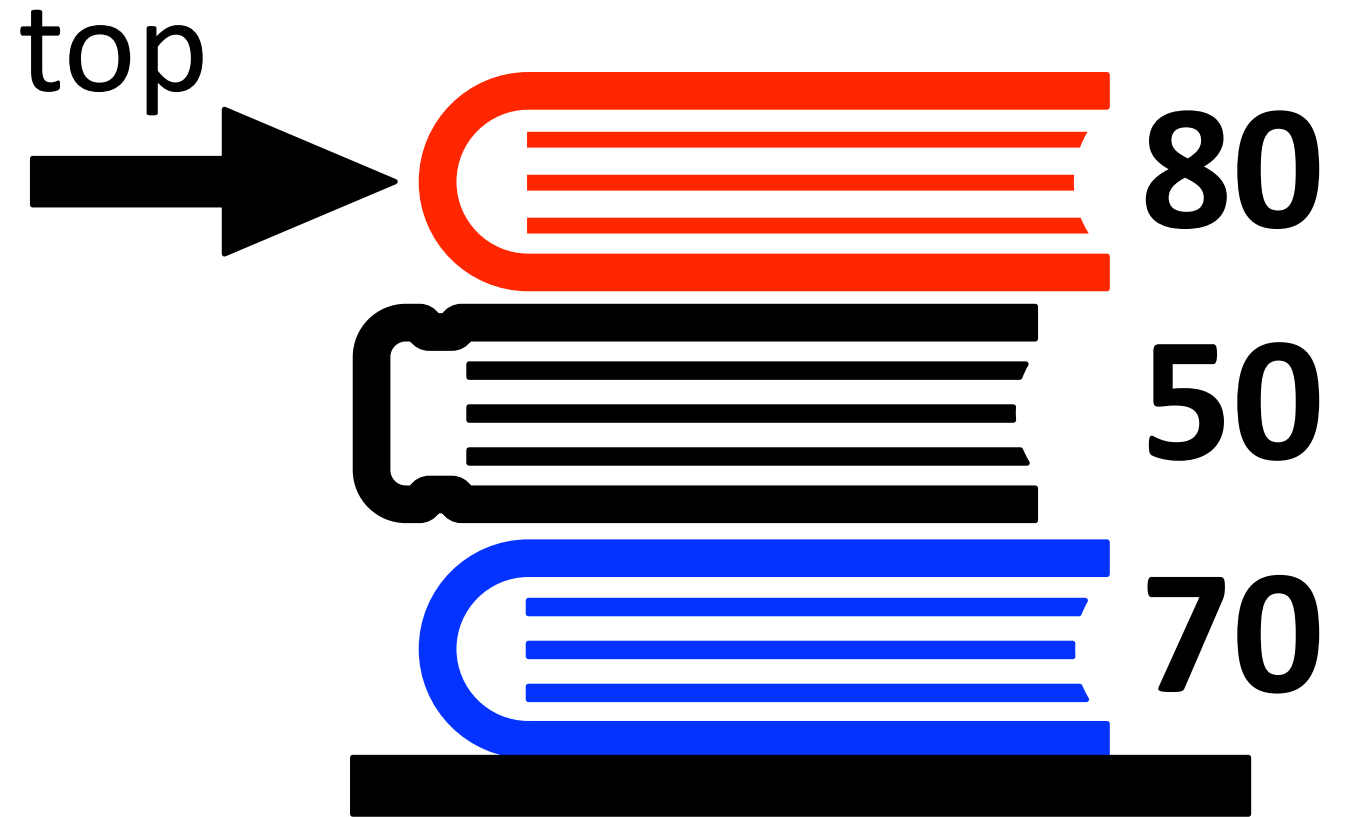
```
stack<int> s
```

```
s.push(70)
```

```
s.push(50)
```

```
s.push(80)
```

```
s.top()
```



**s.pop()** removes value that was pushed in *last*

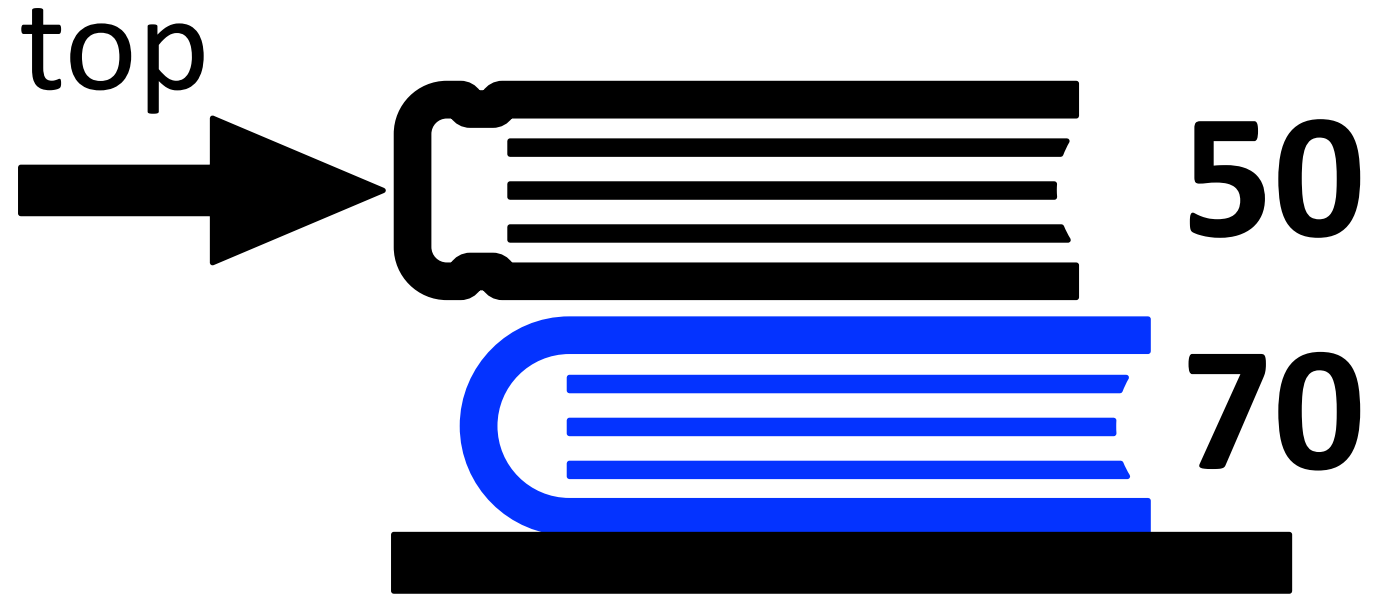
```
stack<int> s
```

```
s.push(70)
```

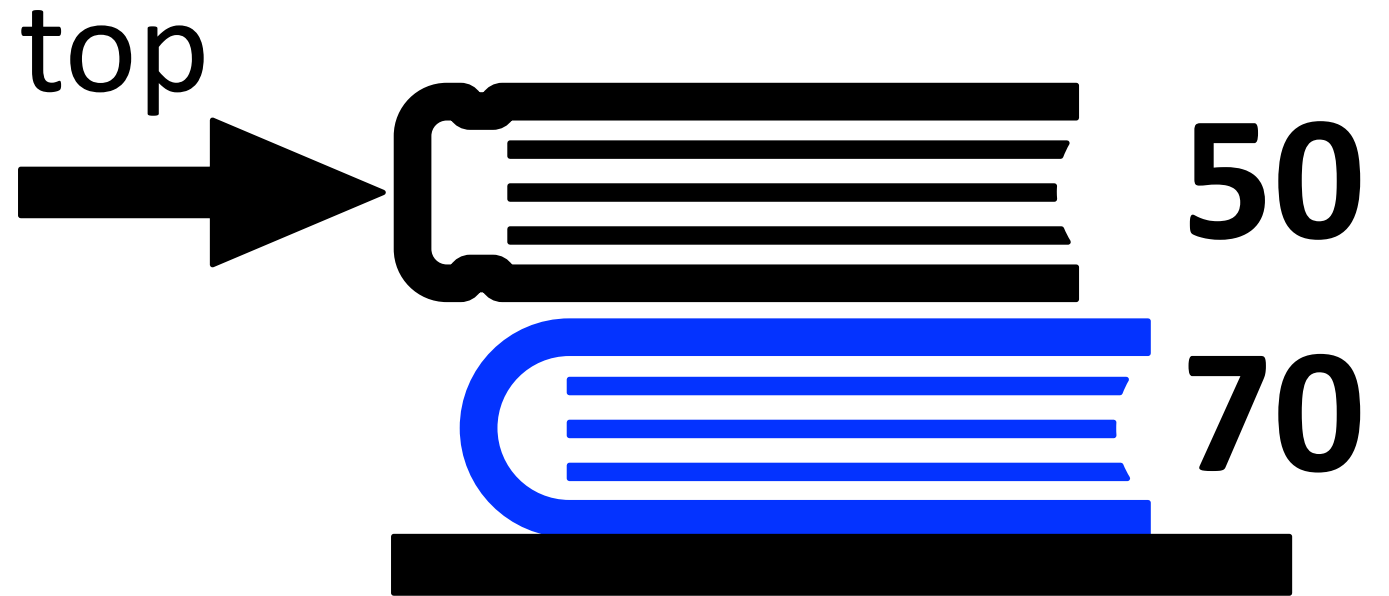
```
s.push(50)
```

```
s.push(80)
```

```
s.top()
```



**s.pop() removes value that was pushed in *last***



**The Last value In is the First value Out (LIFO)**

## The call stack:

```
1  #include <iostream>
2  using namespace std;
3
4  int fact(int n){
5      if(n <= 1) return 1;
6      return n * fact(n - 1);
7  }
8
9  int main() {
10     cout<< fact(4) << endl;
11     return 0;
12 }
```

main

fact(int)

n | int  
4

fact(int)

n | int  
3

fact(int)

n | int  
2

fact(int)

n | int  
1

**The Last value In is the First value Out (LIFO)**

**vector**

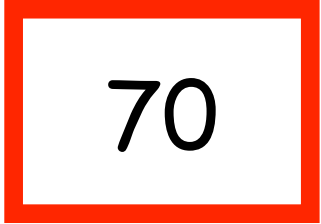
**list**



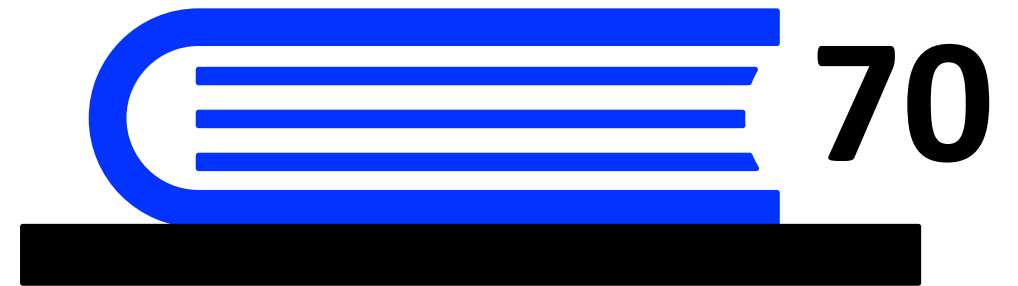
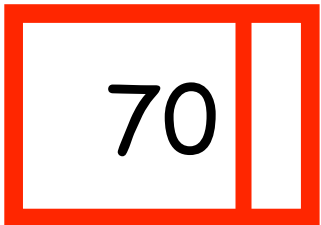
Empty stack

**Stack Abstract Data Type**

**vector**



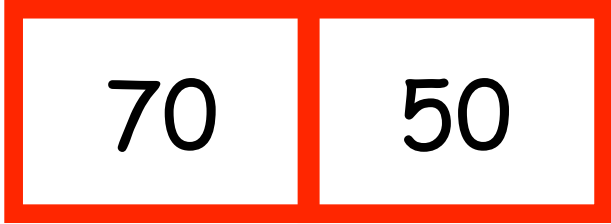
**list**



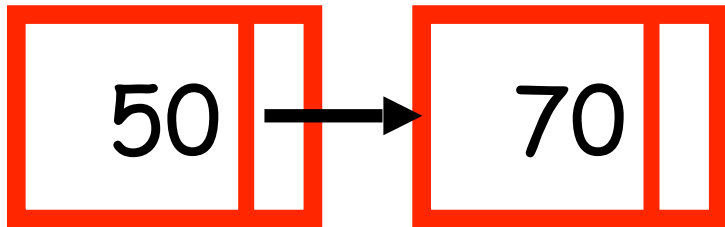
**s.push(70)**

**Stack Abstract Data Type**

**vector**



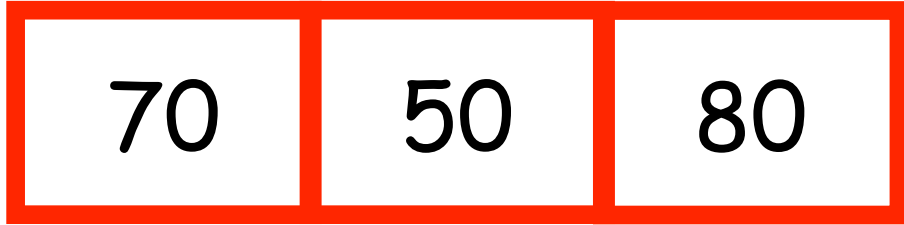
**list**



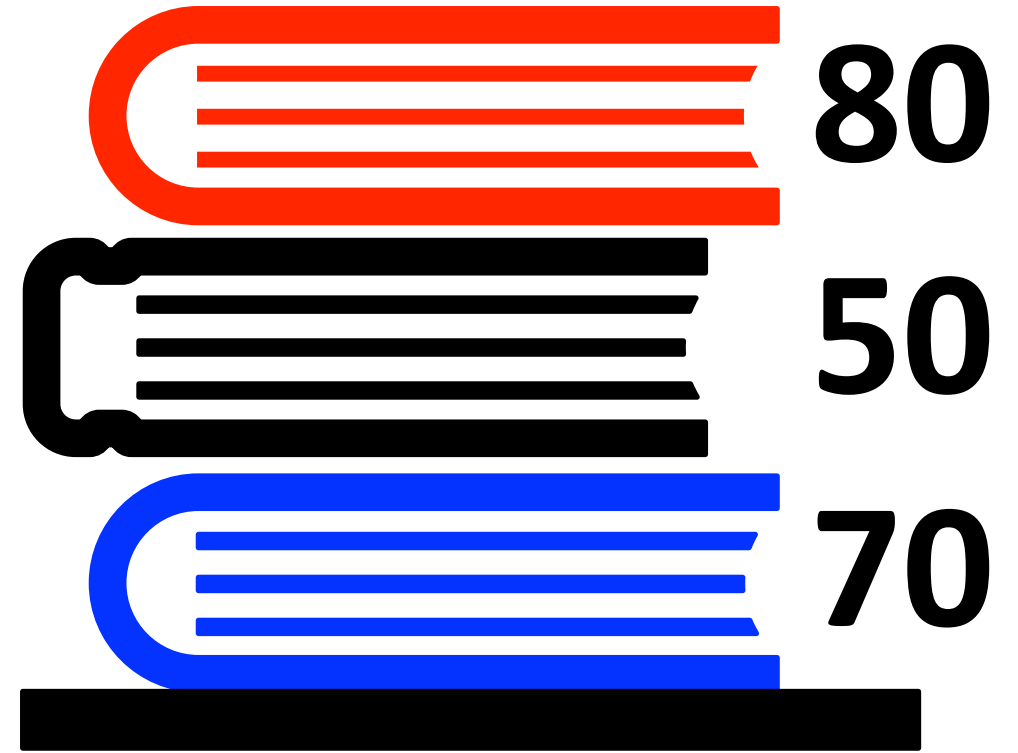
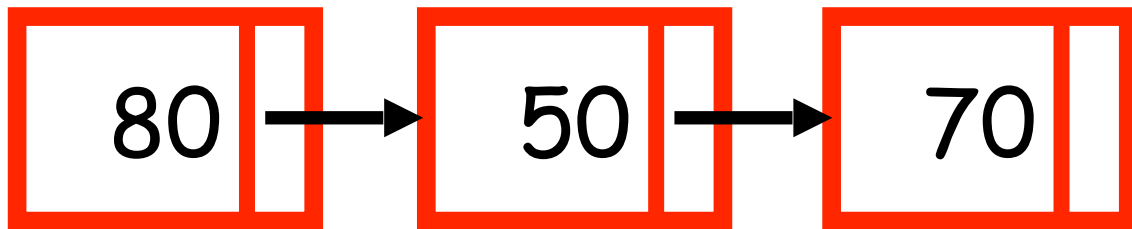
**s.push(50)**

**Stack Abstract Data Type**

**vector**



**list**

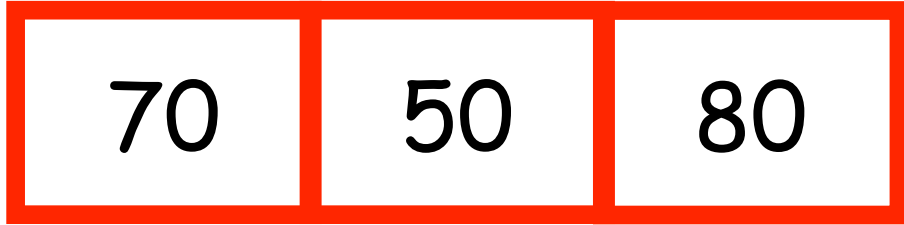


**s.push(80)**

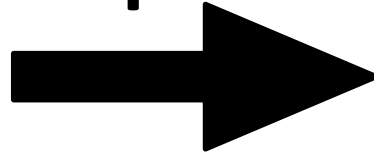
**Stack Abstract Data Type**



**vector**



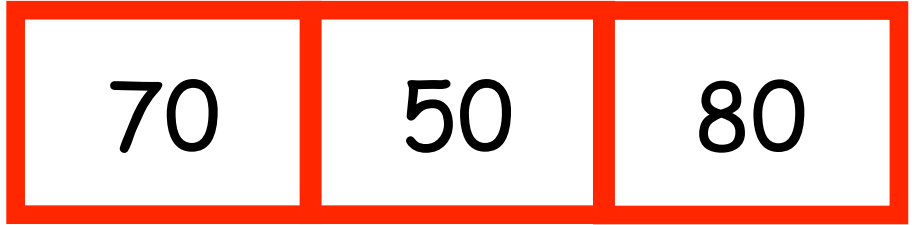
top



The top element is at **index**:

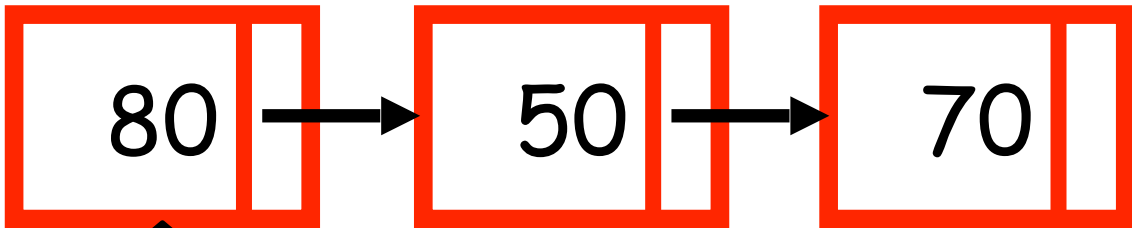
- A. zero
- B. one
- C. `v.size() - 1`
- D. `v.size()`

**vector**

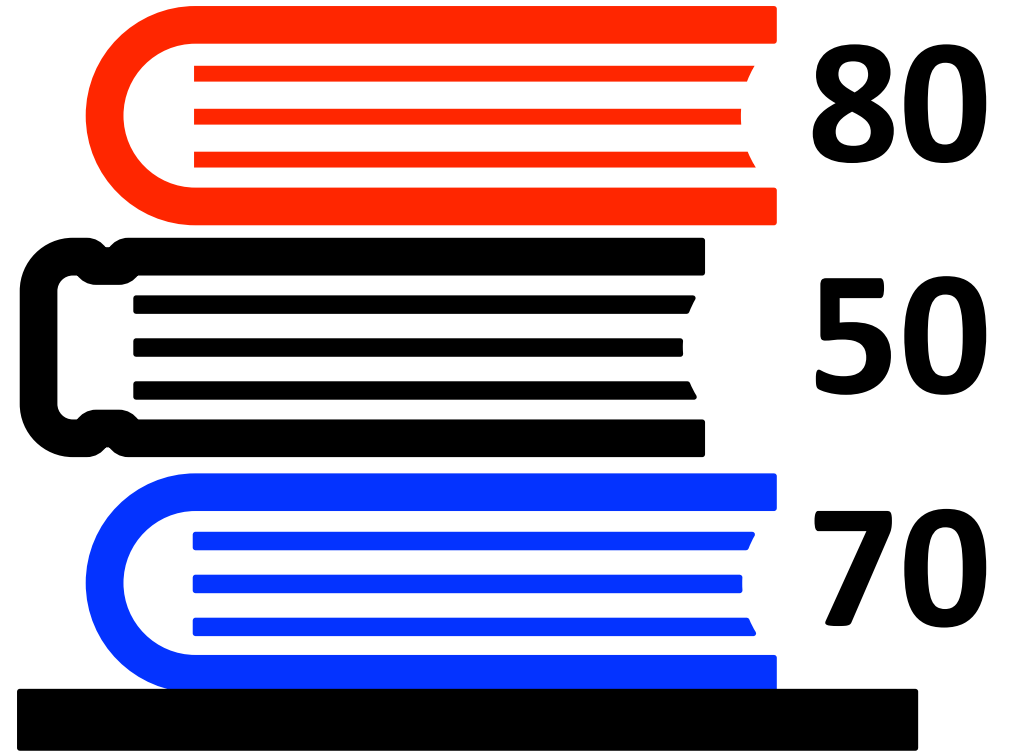


**↑**  
**top**

**list**



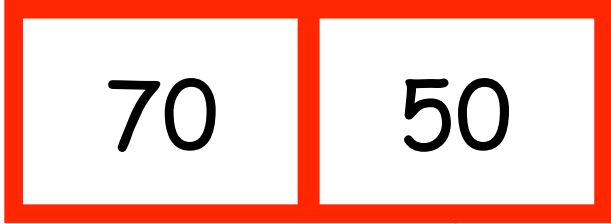
**↑**  
**top**



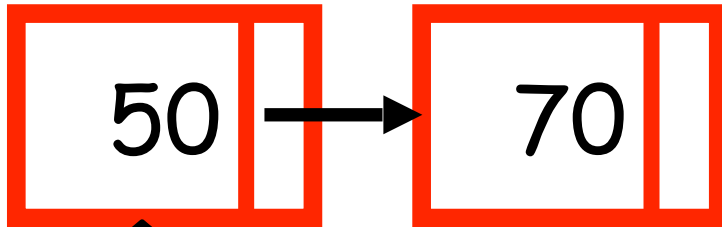
**s.pop()**

**Stack Abstract Data Type**

**vector**



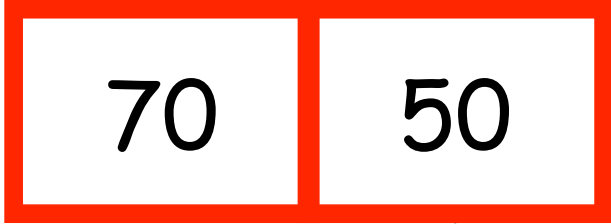
**list**



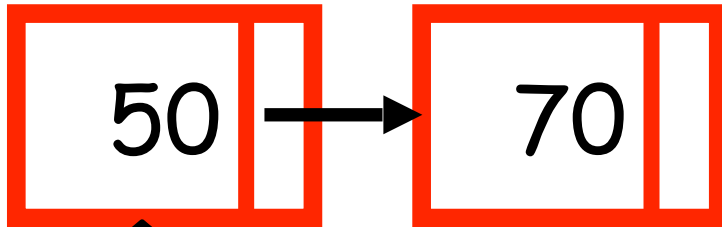
**s.pop()**

**Stack Abstract Data Type**

**vector**



**list**

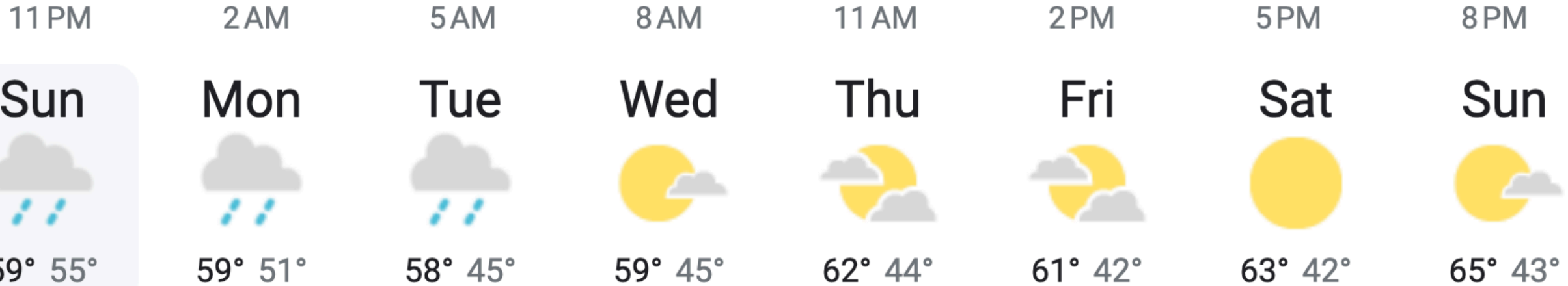


**Stack Abstract Data Type**

Why implement a stack at all?  
After all a stack is a vector or list with a  
**reduced set of operations**

Stack has only three operations: **push()**    **pop()**    **top()**

- Attempt this problem on leetcode
- Spend no more than 30 minutes on it
- We'll discuss the solution on Wednesday



<https://leetcode.com/problems/daily-temperatures/>