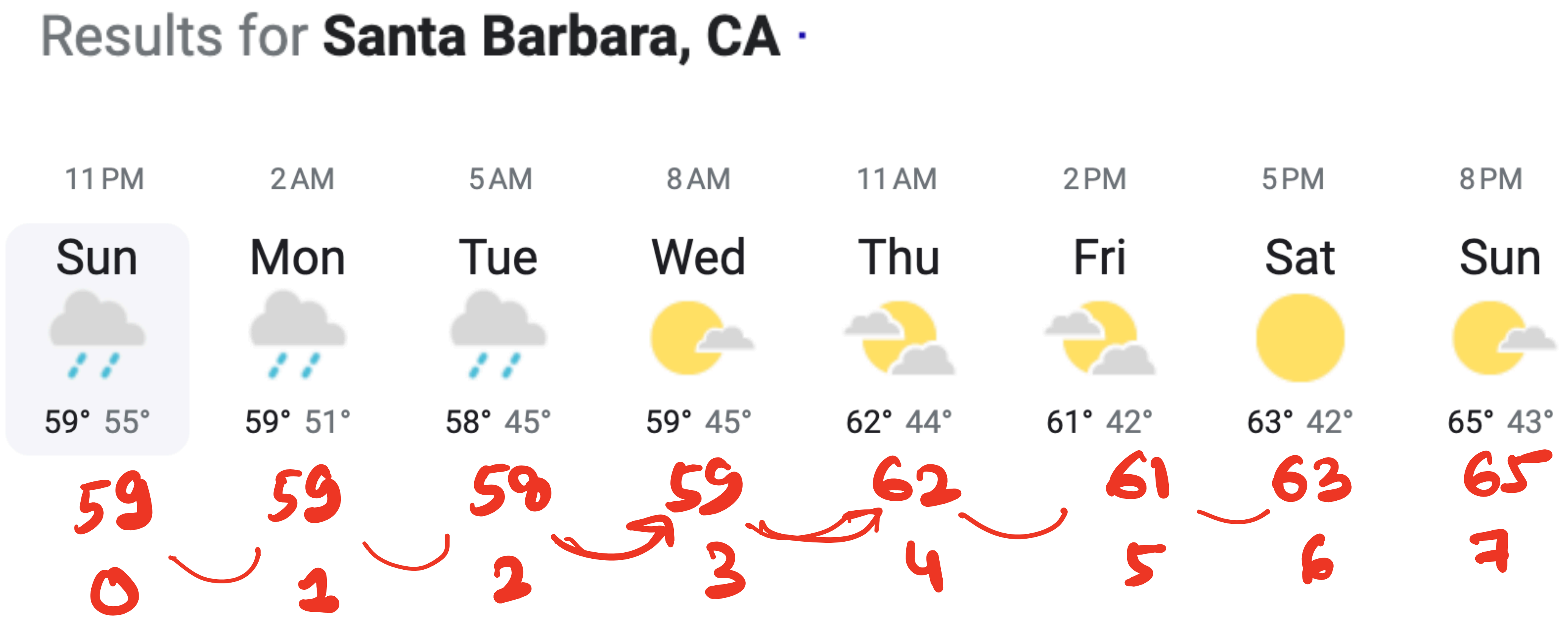
STACK

Problem Solving with Computers-II

```
#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	Mon	Tue	Wed	Thu	Fri	Sat	Sun
59° 55°	59° 51°	58° 45°	59° 45°	62° 44°	61° 42°	63° 42°	65° 43°
59	59	58	59	62	61	63	65
0	1	2	3	4	5	6	7

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

result

4	3	1	1	2	1	1	0
0	1	2	3	4	5	6	7

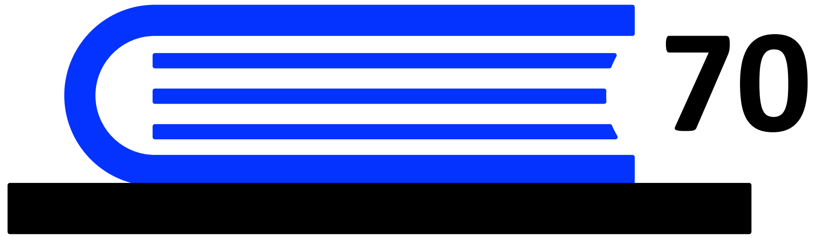
A stack is like a chisel — simple, yet powerful!

stack<int> s

Empty stack

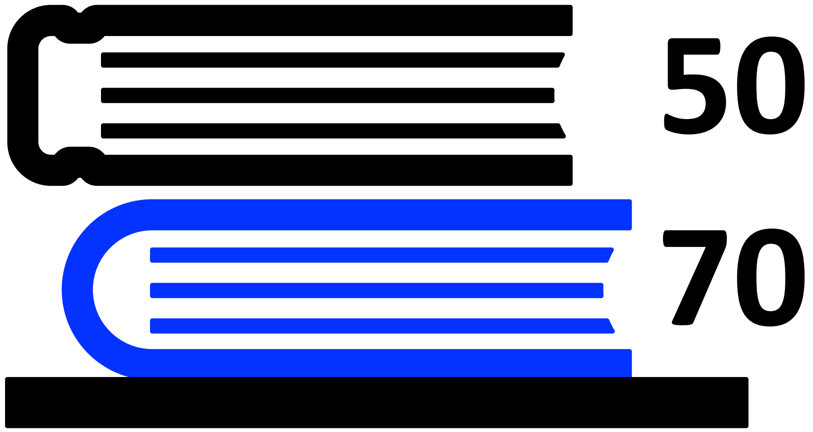
Operations: **push()** **pop()** **top()**

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



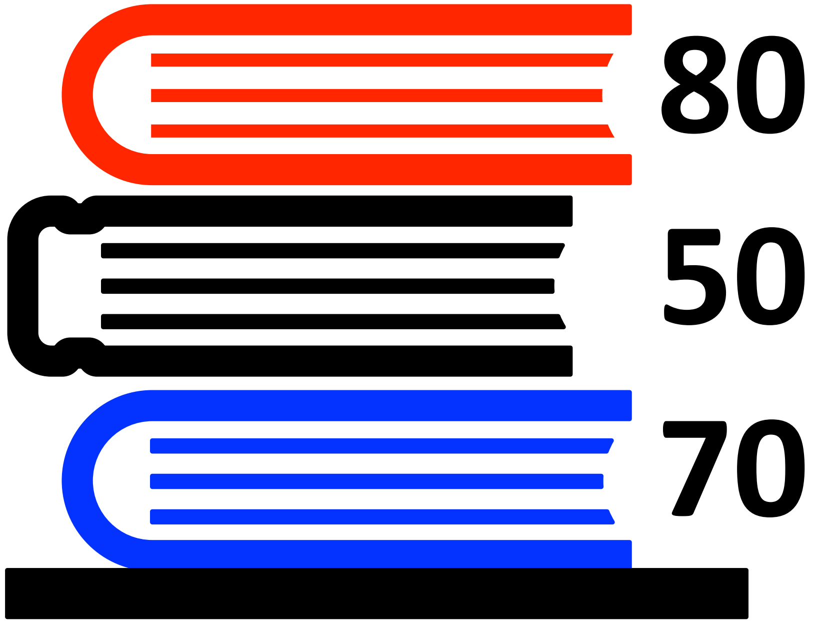
Operations: **push()** pop() top()

```
stack<int> s
s.push(70)
s.push(50)
```



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

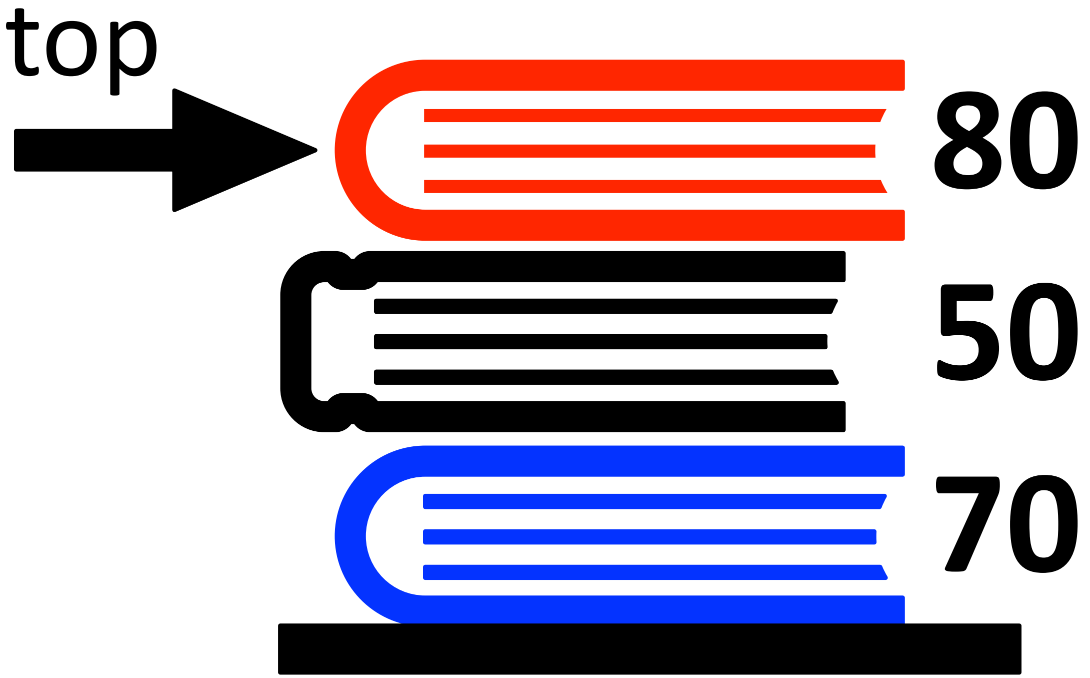
```
stack<int> s
s.push(70)
s.push(50)
s.push(80)
```



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

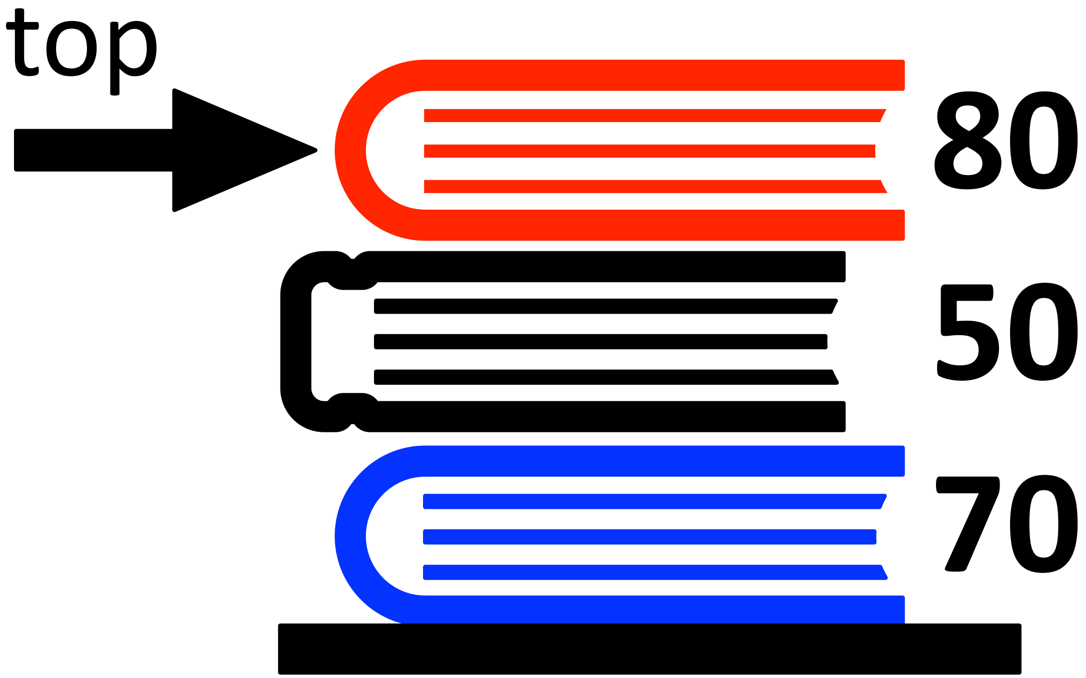
```
stack<int> s
s.push(70)
s.push(50)
s.push(80)
```

s.top() returns 80



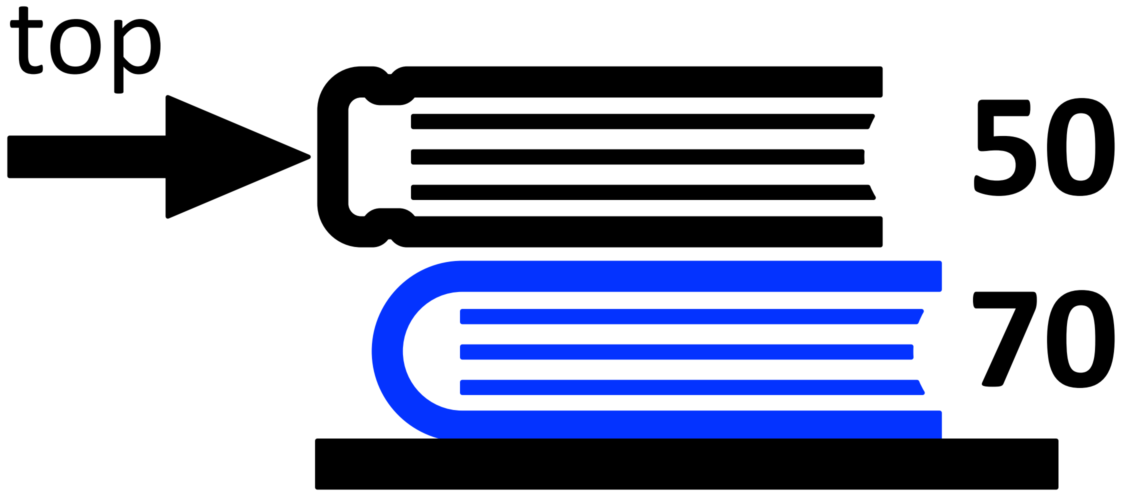
Operations: push() pop() **top()**

```
stack<int> s
s.push(70)
s.push(50)
s.push(80)

s.top()
```

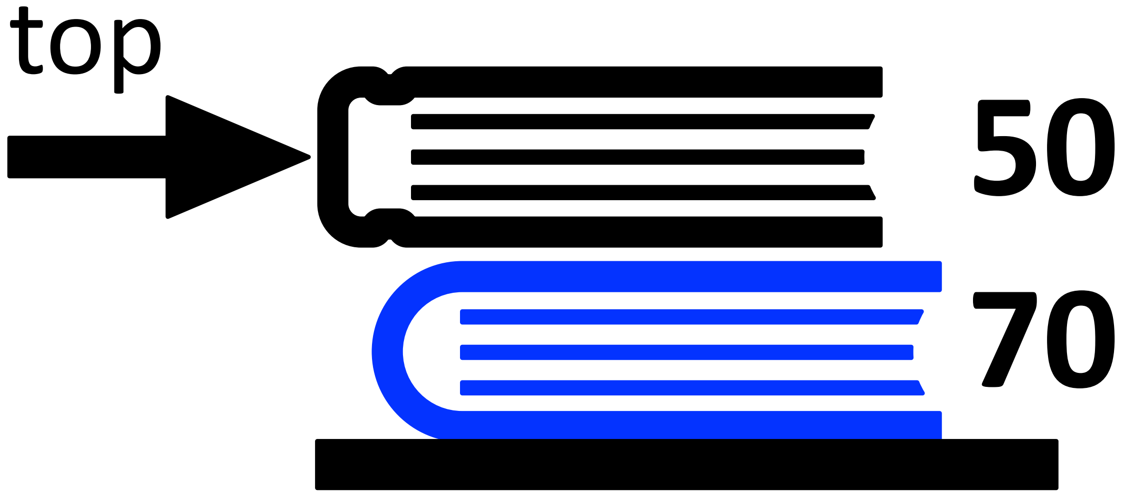


s.pop() removes value that was pushed in *last*

```
stack<int> s
s.push(70)
s.push(50)
s.push(80)

s.top()
```

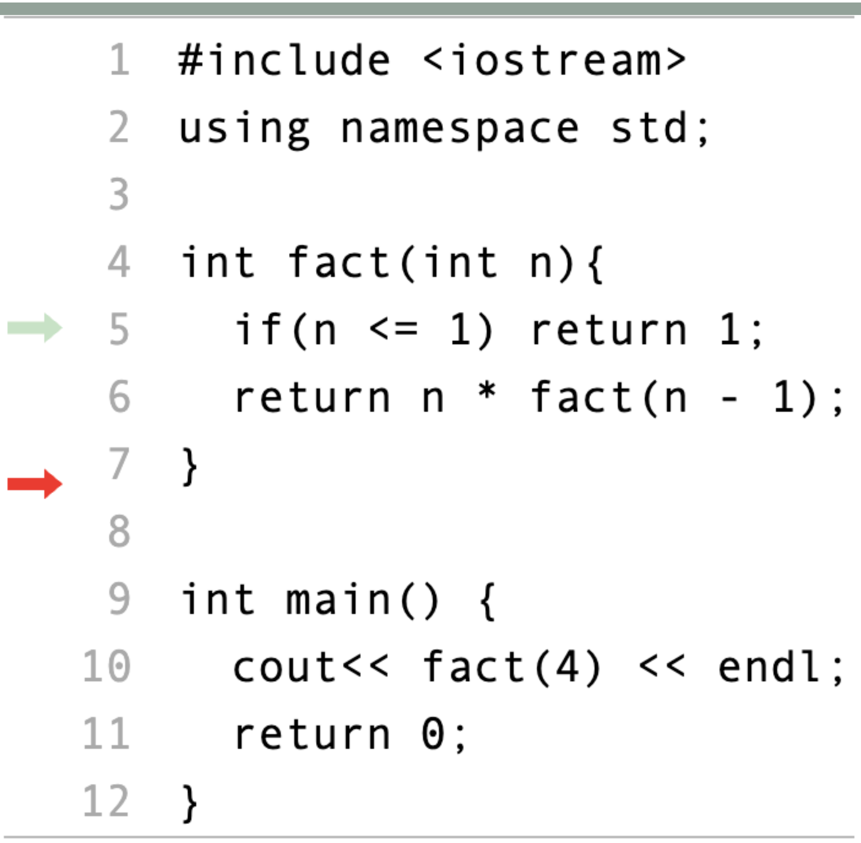


s.pop() removes value that was pushed in *last*

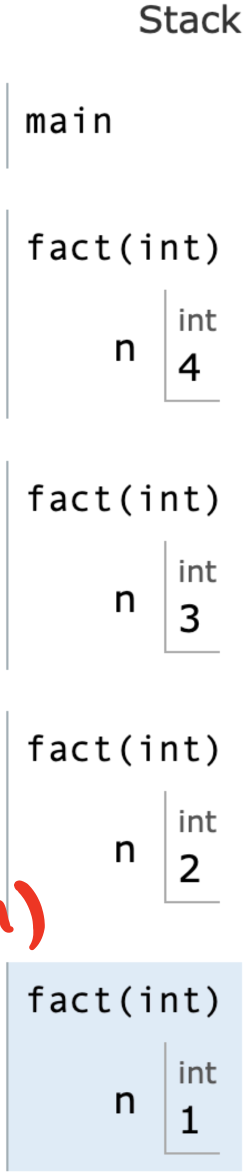


The Last value In is the First value Out (LIFO)

The call stack:



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

int fact(int n){
  if(n <= 1) return 1;
  return n * fact(n - 1);
}

int main() {
  cout<< fact(4) << endl;
  return 0;
}
```



Stack grows this way

Maximum depth of the recursion defines the space complexity O(n)

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

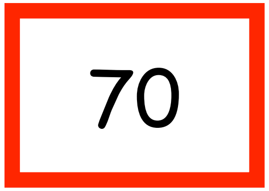
vector

list

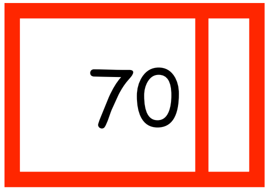
Empty stack

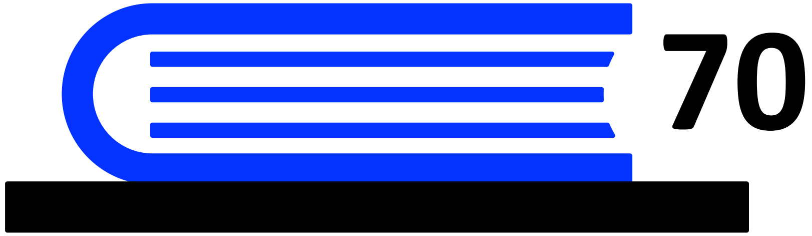
Stack Abstract Data Type

vector



list

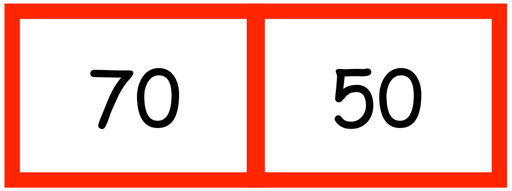




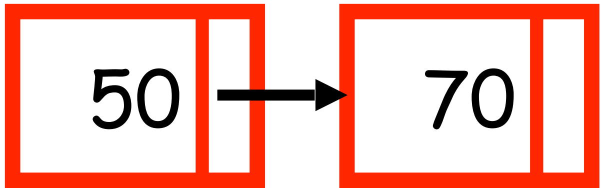
s.push(70)

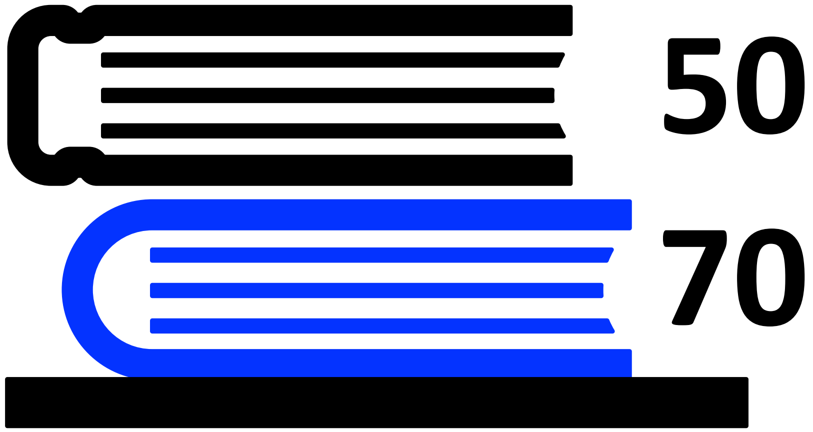
Stack Abstract Data Type

vector



list

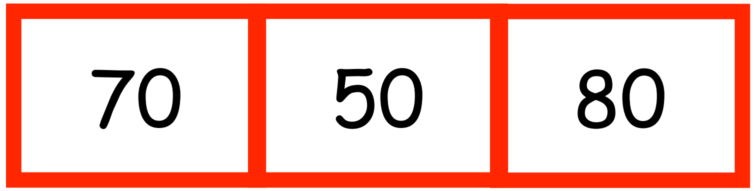




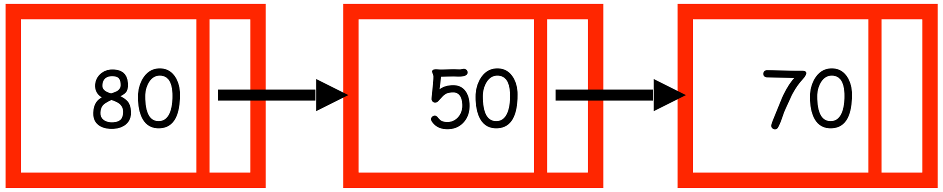
s.push(50)

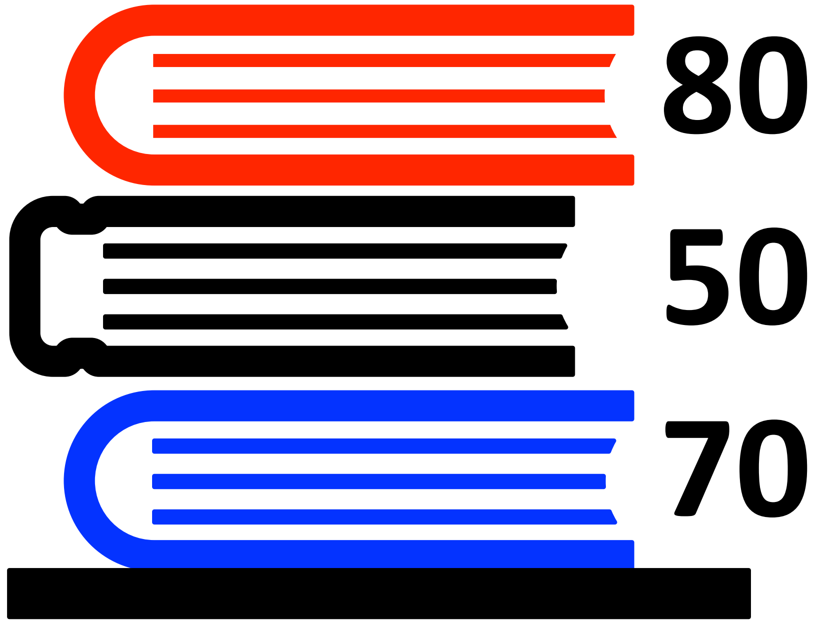
Stack Abstract Data Type

vector



list

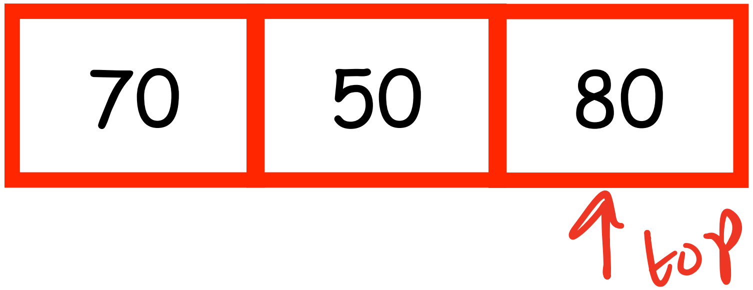


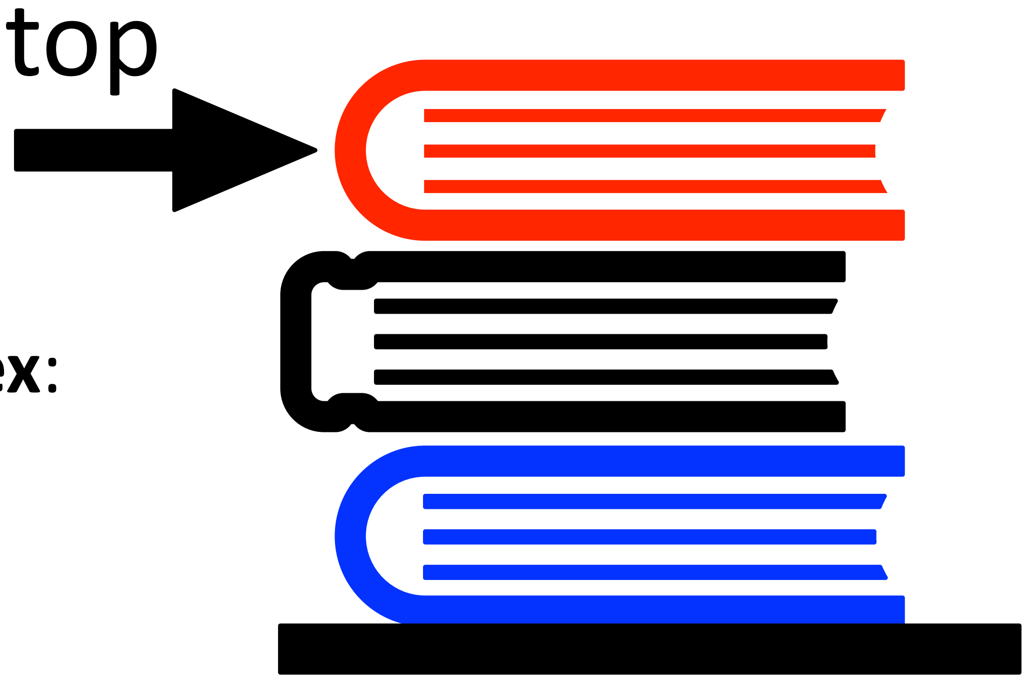


s.push(80)

Stack Abstract Data Type

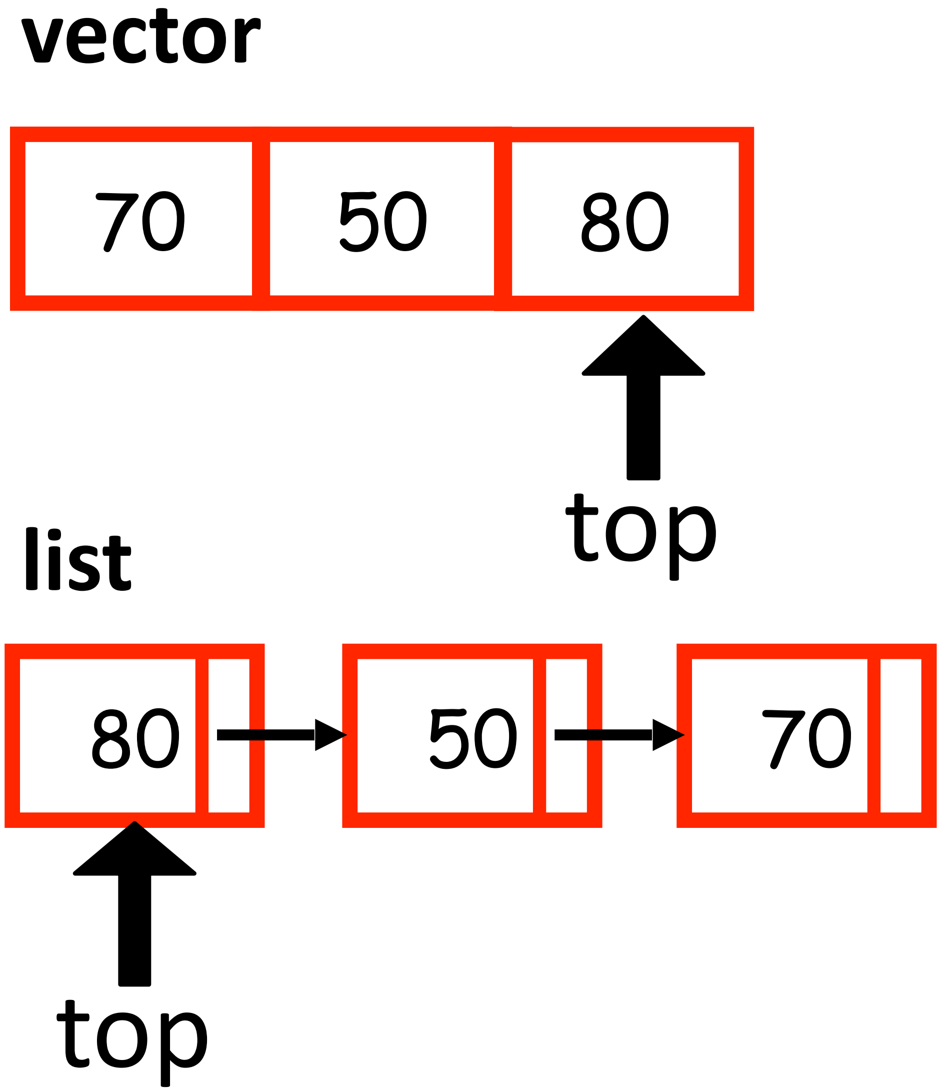
vector

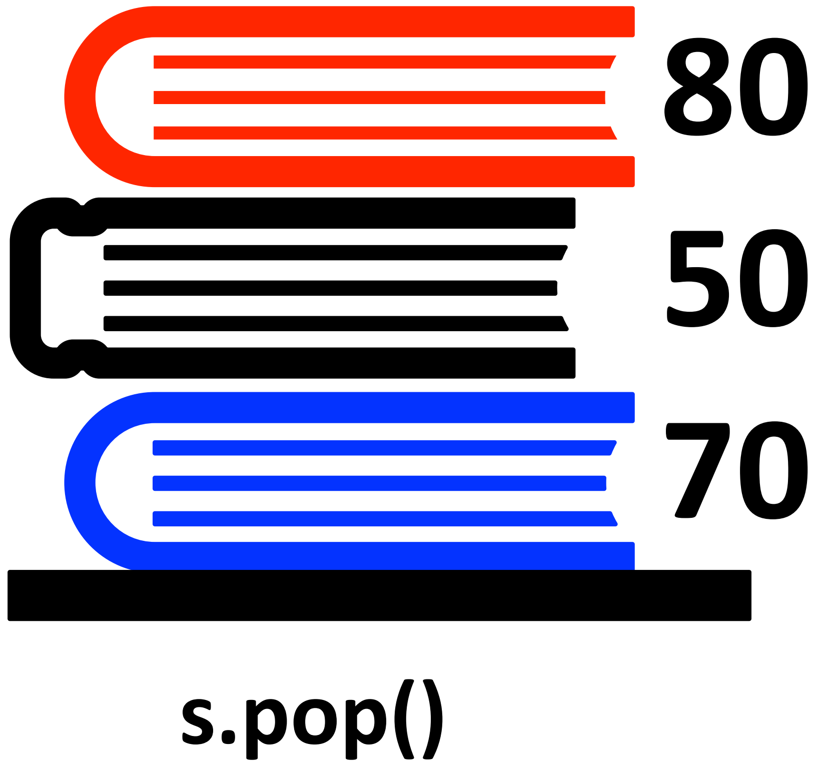




The top element is at **index**:

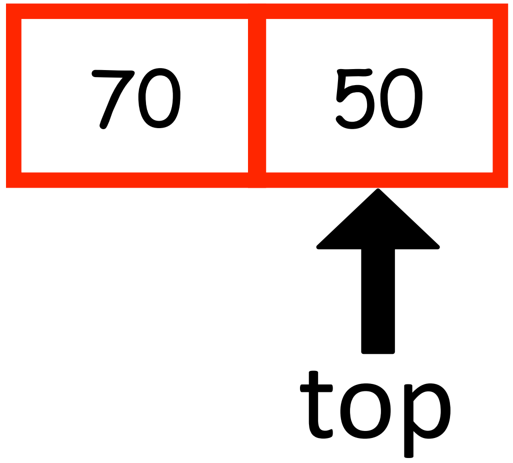
A. zero
B. one
[✓ C.] v.size() - 1
D. v.size()



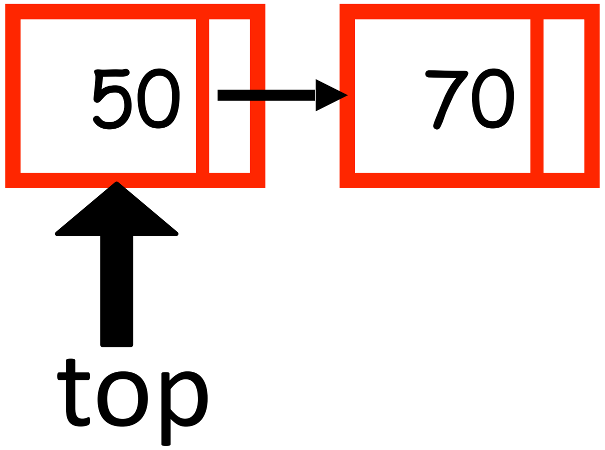


Stack Abstract Data Type

vector



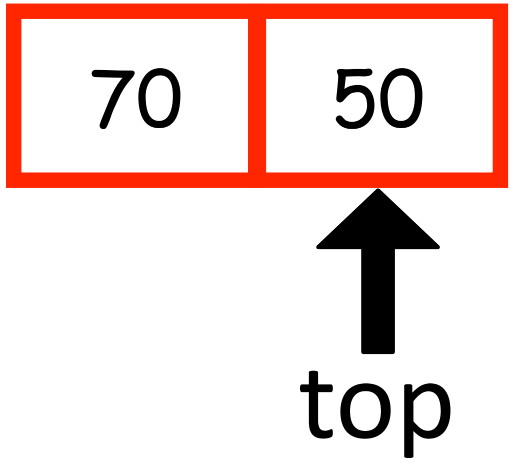
list



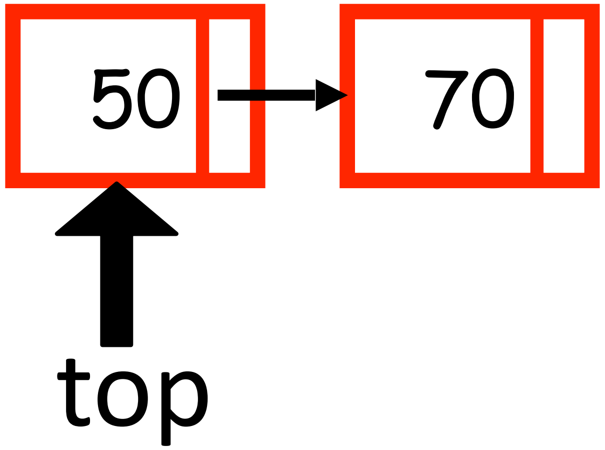
s.pop()

Stack Abstract Data Type

vector



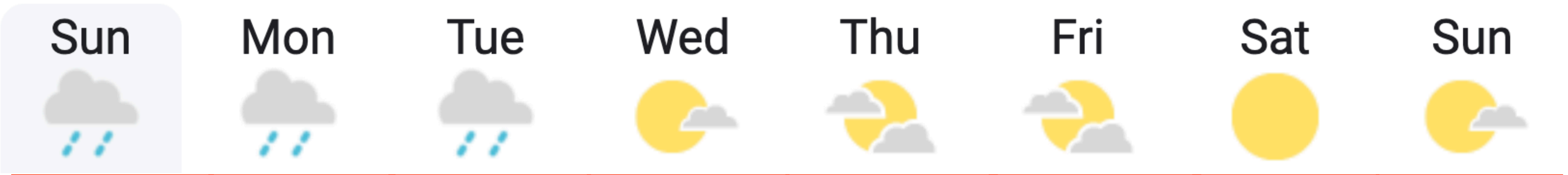
list



Stack Abstract Data Type

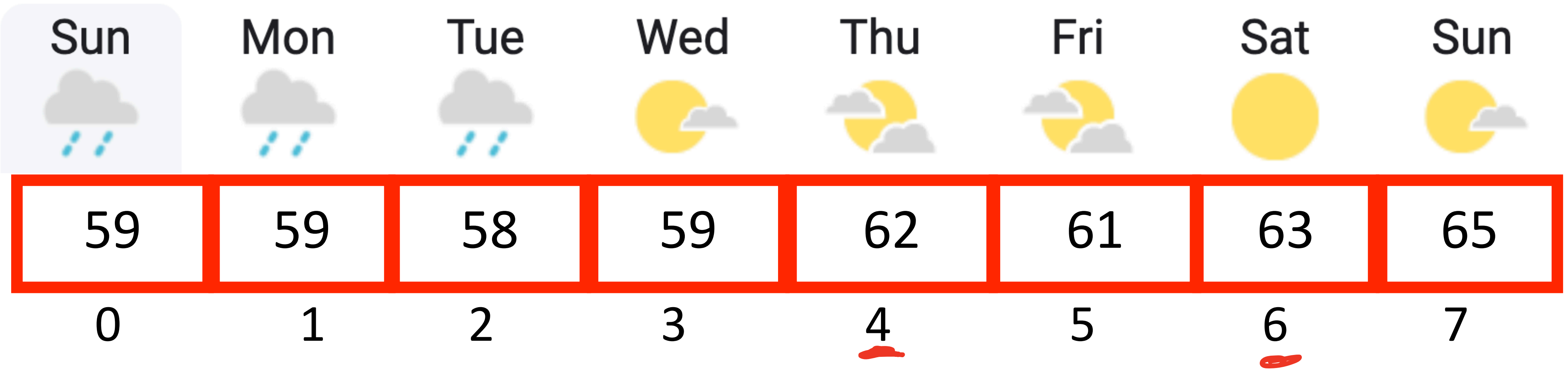
Why implement a stack at all?
After all a stack is a vector or list with a **reduced set of operations**

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



59	59	58	59	62	61	63	65

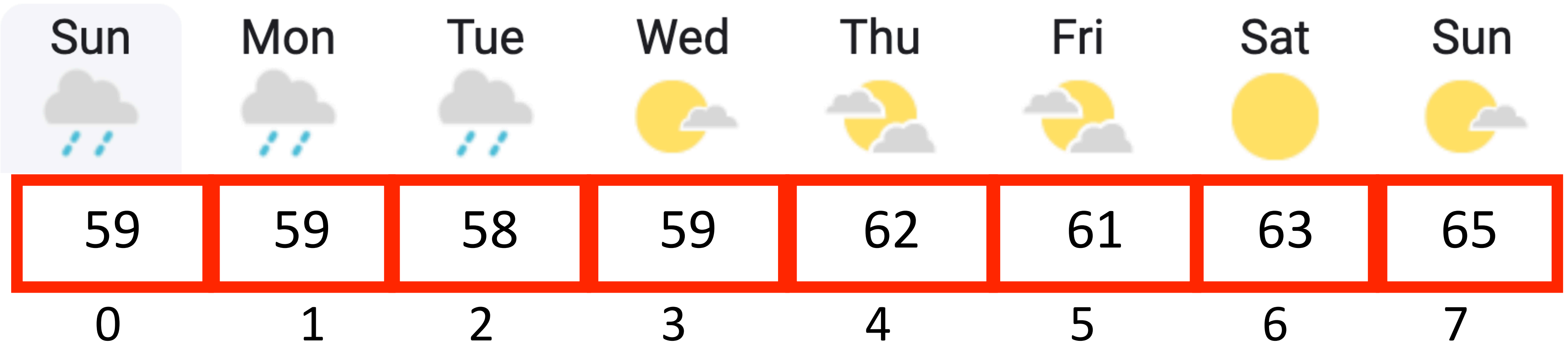
https://leetcode.com/problems/daily-temperatures/



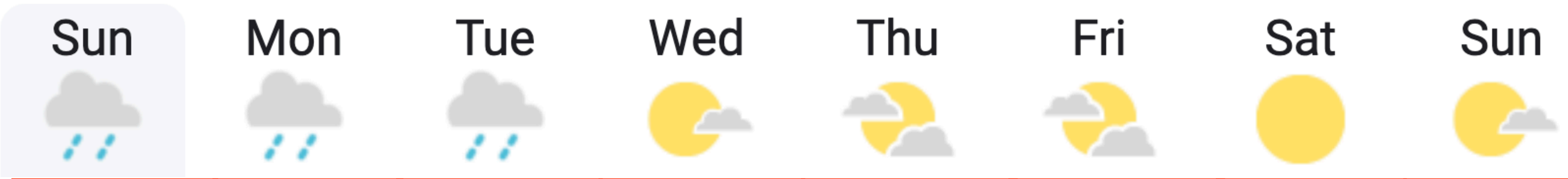
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
59	59	58	59	62	61	63	65
0	1	2	3	4	5	6	7

If we parse the temperatures from right to left, which day(s) should we remember to compute the answer for Day 4 (Thu)?

A. Day 5 because its the most recent day after Day 5
[✓ B.] Day 6 because its the most recent warm day after Day 4
C. Day 7 because its warmest day after Day 4
D. Day 3 because Day 4 is warmer than Day 3



If we parse the temperatures from right to left, every day we encounter could be a potential answer (for some preceding day) — **remember potential answers in a stack!**

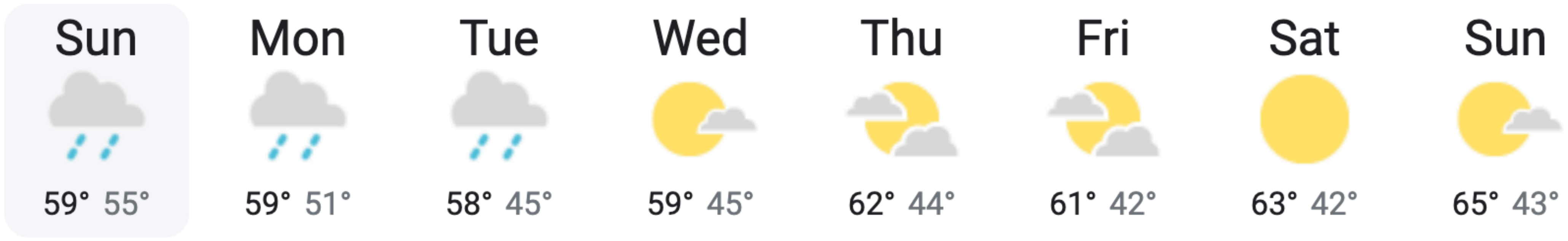


Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
59	59	58	59	62	61	63	65
0	1	2	3	4	5	6	7

However, some values become stale (i.e. they are no longer a potential answer)

How can we detect stale values in the stack and permanently remove them?

- **Attempt a different solution to this problem on leetcode**
- **Spend no more than 30 minutes on it**
- **Discuss your solutions with the course staff in office hours**



Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
59° 55°	59° 51°	58° 45°	59° 45°	62° 44°	61° 42°	63° 42°	65° 43°

A stack is useful for keeping track of history information where computation only depends on the most recent information !!

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