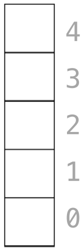
Stacks

We can use an N-element array A, with elements stored from $A[0]$ to $A[t]$, where t is an integer, initialized to -1, that gives the index of the top element in A.

Insertion and removal:

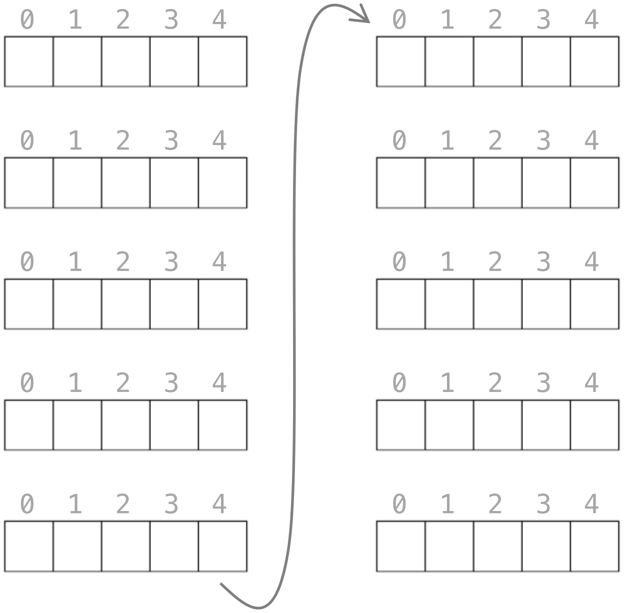
```
Algorithm push($obj$):
    if $(t + 1) = N$ then
        return an error that the stack is full
    $t \leftarrow t + 1$
    $A[t] \leftarrow obj$

Algorithm pop():
    if $t < 0$ then
        return an error that the stack is empty
    $e \leftarrow A[t]$
    $t \leftarrow t - 1$
    return $e$
```



```
s.push(7);
s.push(2);
s.pop();
s.push(9);
s.push(1);
s.push(4);
s.push(8);
s.pop()
s.pop();
s.push(5);
```

Queues

1) Trace the following code:



```
q.enqueue(2);
q.enqueue(4);
q.enqueue(6);
q.dequeue();
q.enqueue(8);
q.dequeue()
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(1);
```

2) Write pseudocode for the enqueue operation

3) Write pseudocode for the dequeue operation

Key ideas: