# Week 9

## Stacks, Queueueueueuueeeues

# Stack

# Queue

# Stack

# Queue

# Stack

FILO: First In, Last Out

# Queue

# Stack

FILO: First In, Last Out

# Queue

FIFO: First In, First Out

# Stack

FILO: First In, Last Out

**Push**: Add to top

**Pop**: Remove from top

**Peek**: Look at top, don’t remove

# Queue

FIFO: First In, First Out

# Stack

FILO: First In, Last Out

**Push**: Add to top
**Pop**: Remove from top
**Peek**: Look at top, don’t remove

# Queue

FIFO: First In, First Out

**Enqueue**: Add to back
**Dequeue**: Remove from front
**Peek**: Look at front, don’t remove

# Stack

FILO: First In, Last Out

**Push**: Add to top

**Pop**: Remove from top

**Peek**: Look at top, don't remove

# Queue

FIFO: First In, First Out

~~**Enqueue**~~: Add to back

~~**Dequeue**~~: Remove from front

~~**Peek**~~: Look at front, don't remove

# Stack

FILO: First In, Last Out

**Push**: Add to top
**Pop**: Remove from top
**Peek**: Look at top, don't remove

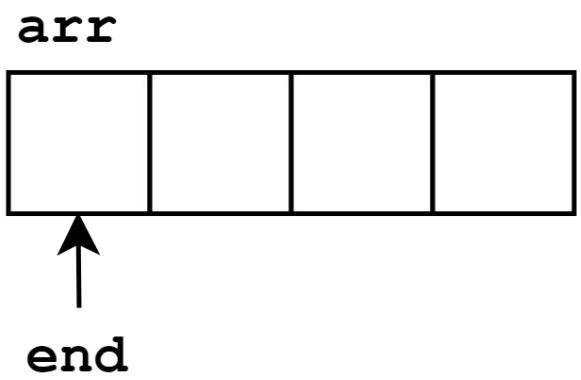
# Queue

FIFO: First In, First Out

**Offer**: Add to back
**Poll**: Remove from front
**Element**: Look at front

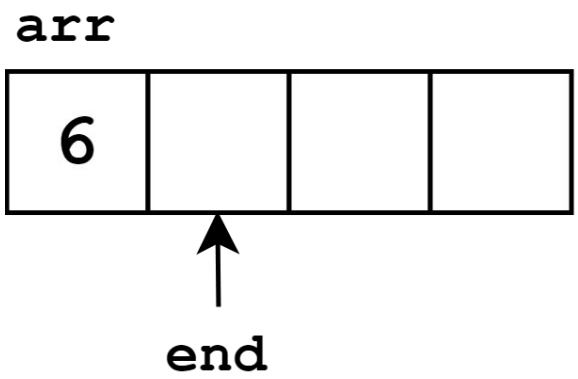
# Implementations

# Stacks with Arrays



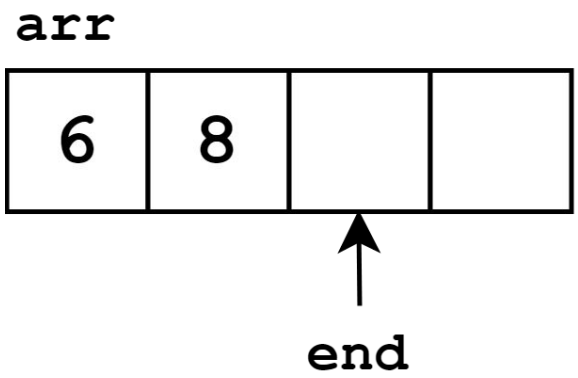
# Stacks with Arrays

```
stack.push(6)
```



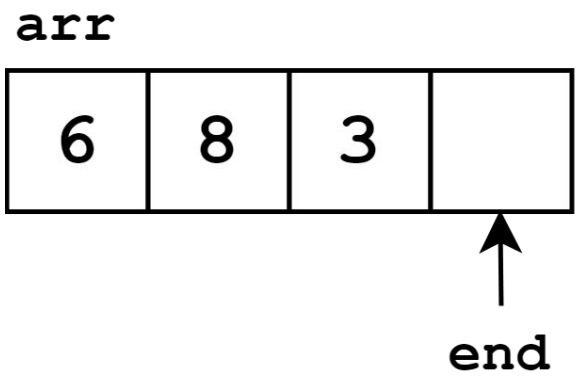
# Stacks with Arrays

```
stack.push(6)
stack.push(8)
```



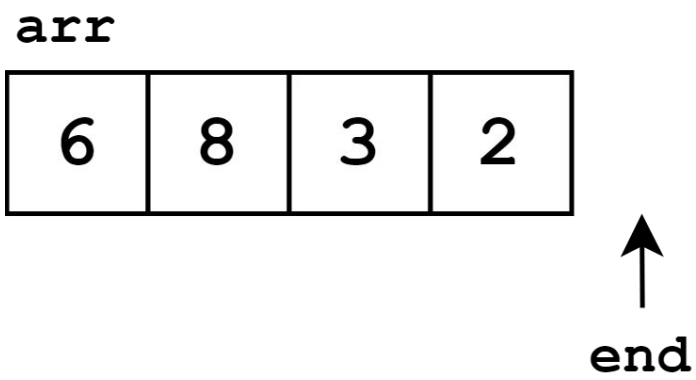
# Stacks with Arrays

```
stack.push(6)
stack.push(8)
stack.push(3)
```



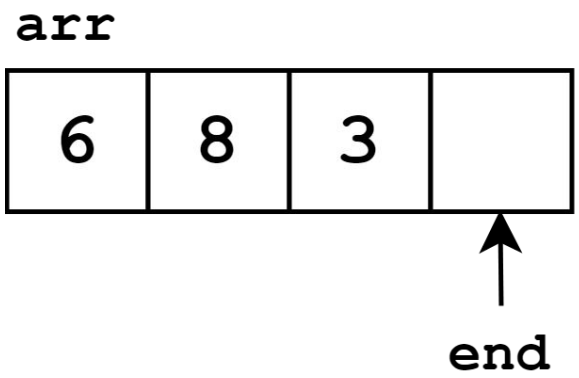
# Stacks with Arrays

```
stack.push(6)
stack.push(8)
stack.push(3)
stack.push(2)
```



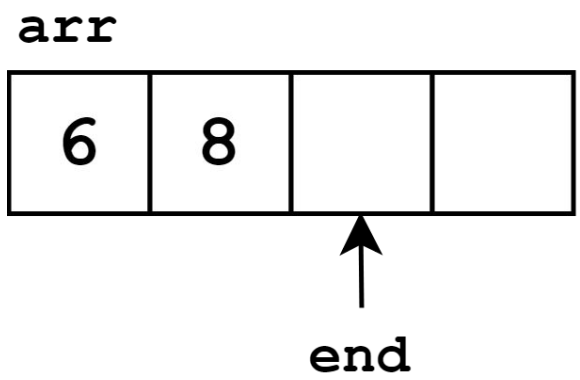
# Stacks with Arrays

```
stack.push(6)
stack.push(8)
stack.push(3)
stack.push(2)
stack.pop() //2
```



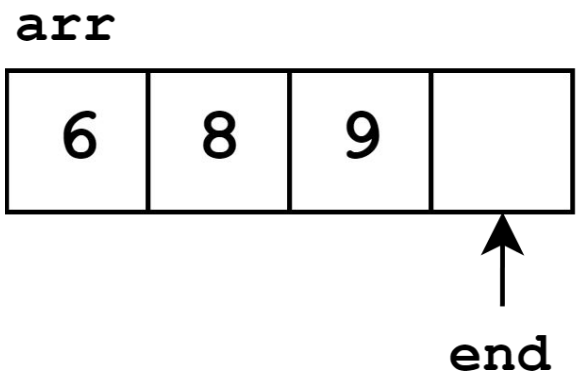
# Stacks with Arrays

```
stack.push(6)
stack.push(8)
stack.push(3)
stack.push(2)
stack.pop() //2
stack.pop() //3
```



# Stacks with Arrays

```
stack.push(6)
stack.push(8)
stack.push(3)
stack.push(2)
stack.pop() //2
stack.pop() //3
stack.push(9)
```



# Stacks with Arrays

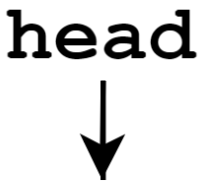
**Push(E element)**

```
arr[end] = element
end+=1
```

**Pop()**

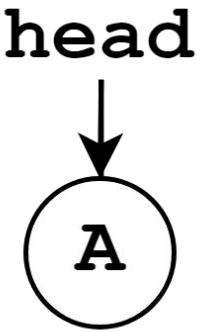
```
element = arr[end-1]
end-=1
return element
```

# Stacks with Linked Lists



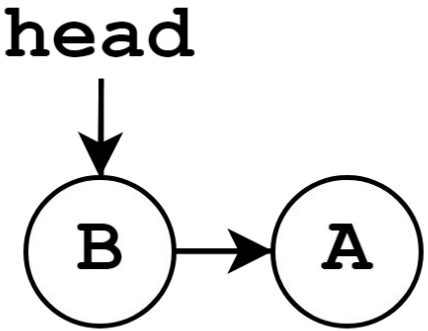
# Stacks with Linked Lists

```
stack.push('A')
```



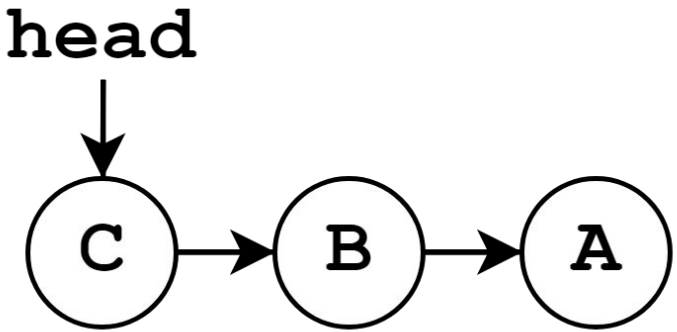
# Stacks with Linked Lists

```
stack.push('A')
stack.push('B')
```



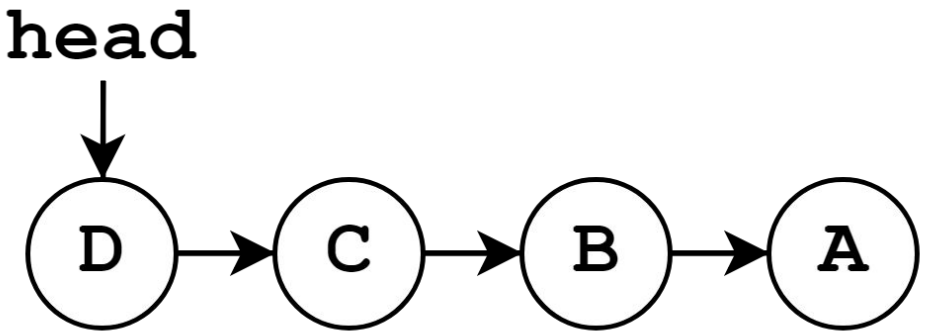
# Stacks with Linked Lists

```
stack.push('A')
stack.push('B')
stack.push('C')
```



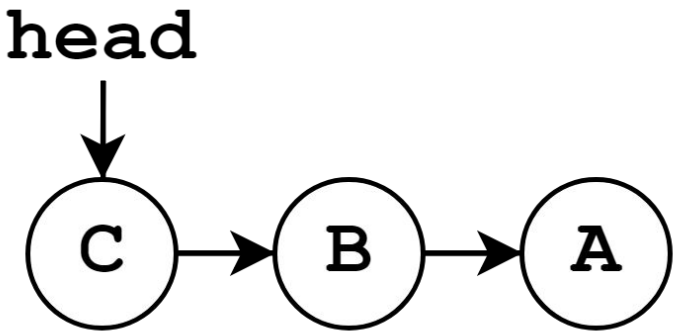
# Stacks with Linked Lists

```
stack.push('A')
stack.push('B')
stack.push('C')
stack.push('D')
```



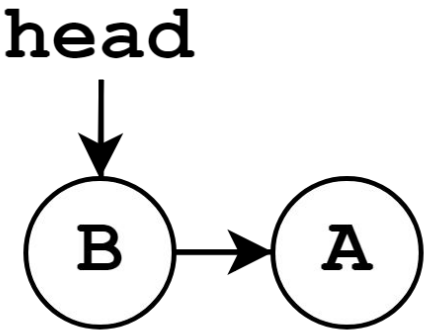
# Stacks with Linked Lists

```
stack.push('A')
stack.push('B')
stack.push('C')
stack.push('D')
stack.pop() //D
```



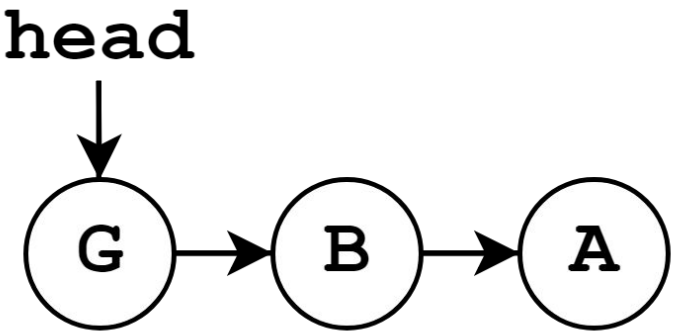
# Stacks with Linked Lists

```
stack.push('A')
stack.push('B')
stack.push('C')
stack.push('D')
stack.pop() //D
stack.pop() //C
```



# Stacks with Linked Lists

```
stack.push('A')
stack.push('B')
stack.push('C')
stack.push('D')
stack.pop() //D
stack.pop() //C
stack.push('G')
```



# Stacks with Linked Lists

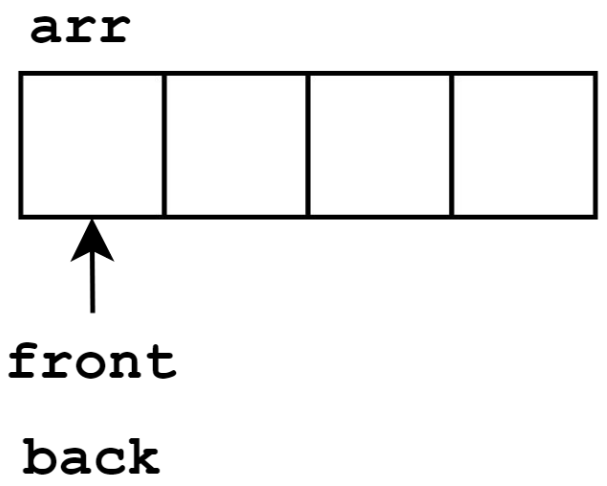
**Push(E data)**

```
n = new Node(data)
n.next = head
head = n
```

**Pop()**

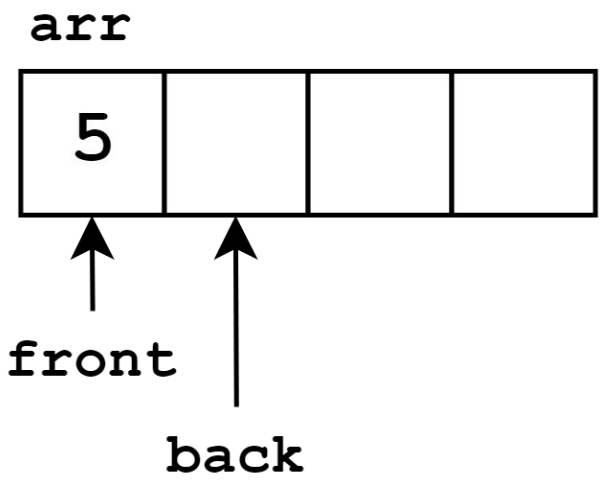
```
data = head.data
head = head.next
return data
```

# Queues with Arrays



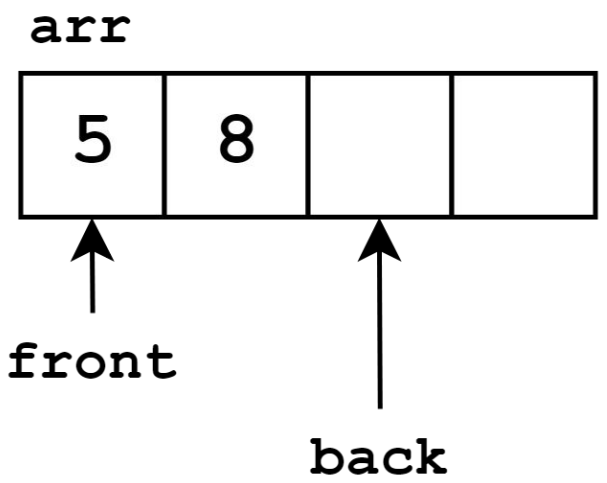
# Queues with Arrays

```
queue.offer(5)
```



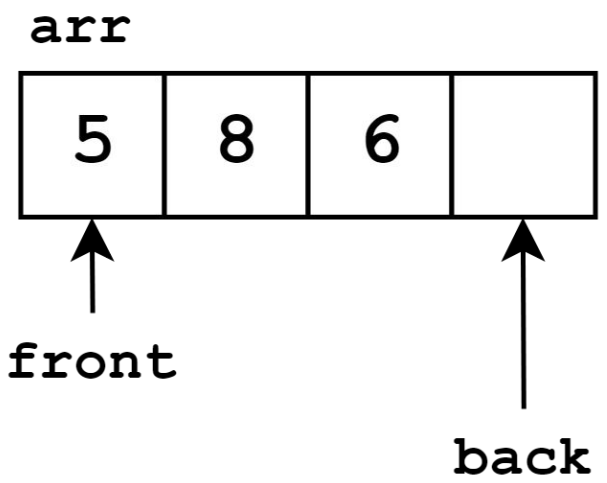
# Queues with Arrays

```
queue.offer(5)
queue.offer(8)
```



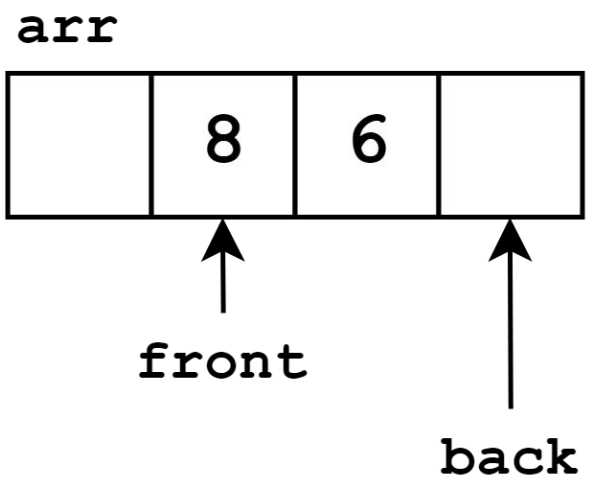
# Queues with Arrays

```
queue.offer(5)
queue.offer(8)
queue.offer(6)
```



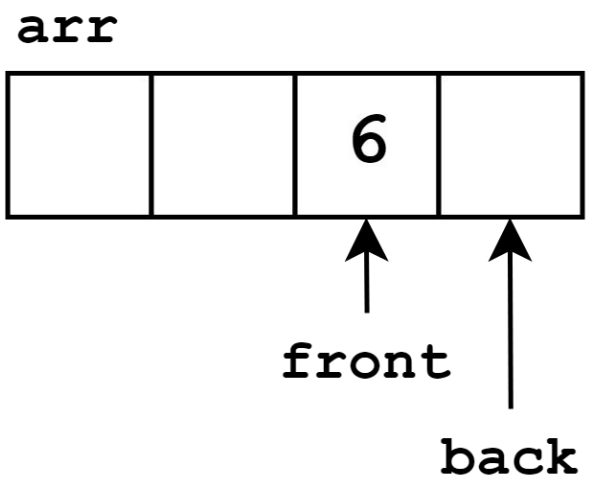
# Queues with Arrays

```
queue.offer(5)
queue.offer(8)
queue.offer(6)
queue.poll() //5
```



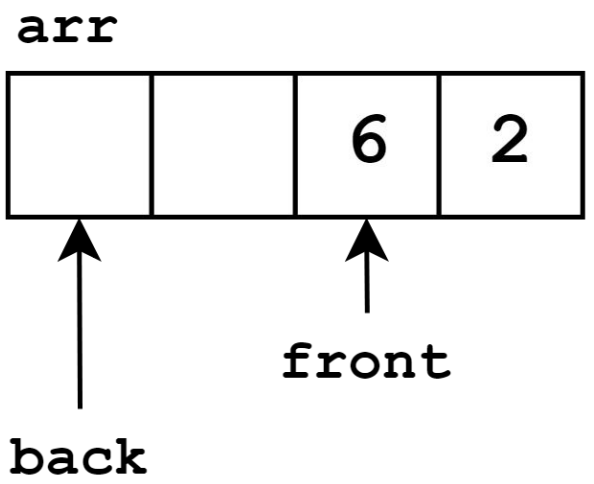
# Queues with Arrays

```
queue.offer(5)
queue.offer(8)
queue.offer(6)
queue.poll() //5
queue.poll() //8
```



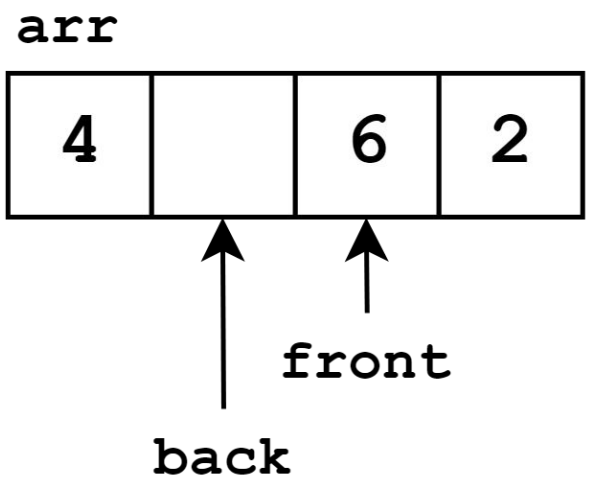
# Queues with Arrays

```
queue.offer(5)
queue.offer(8)
queue.offer(6)
queue.poll() //5
queue.poll() //8
queue.offer(2)
```



# Queues with Arrays

```
queue.offer(5)
queue.offer(8)
queue.offer(6)
queue.poll() //5
queue.poll() //8
queue.offer(2)
queue.offer(4)
```



# Queues with Arrays

**Offer(E data)**

```
arr[back] = data
n = arr.length
back = (back+1)%n
```

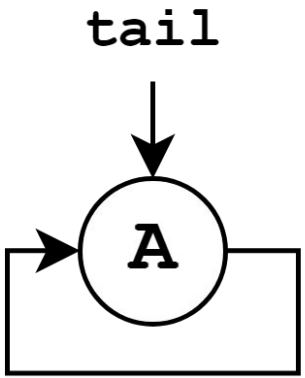
**Poll()**

```
data = arr[front]
n = arr.length
front = (front+1)%n
return data
```

# Queues with Linked Lists

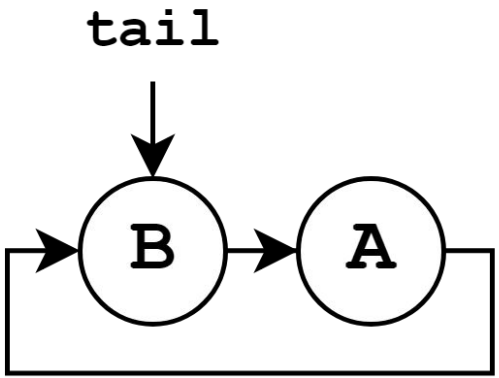
# Queues with Linked Lists

```
queue.offer('A')
```



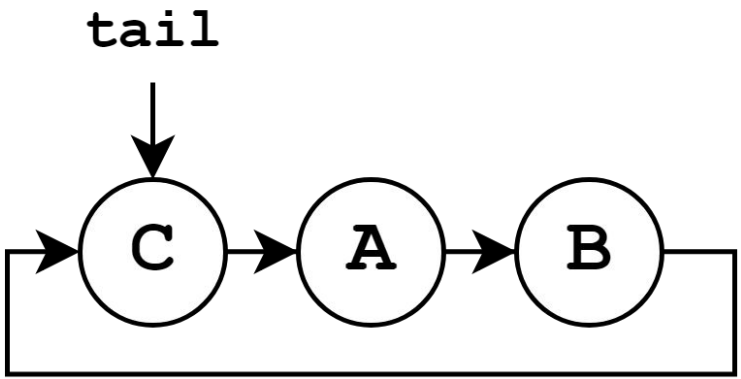
# Queues with Linked Lists

```
queue.offer('A')
queue.offer('B')
```



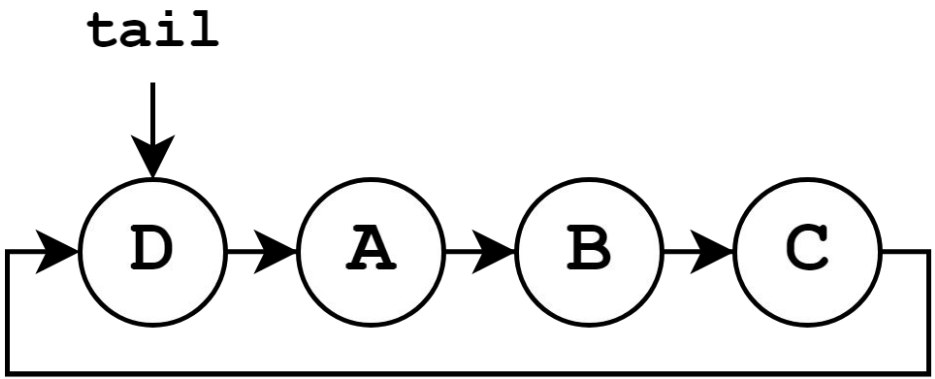
# Queues with Linked Lists

```
queue.offer('A')
queue.offer('B')
queue.offer('C')
```



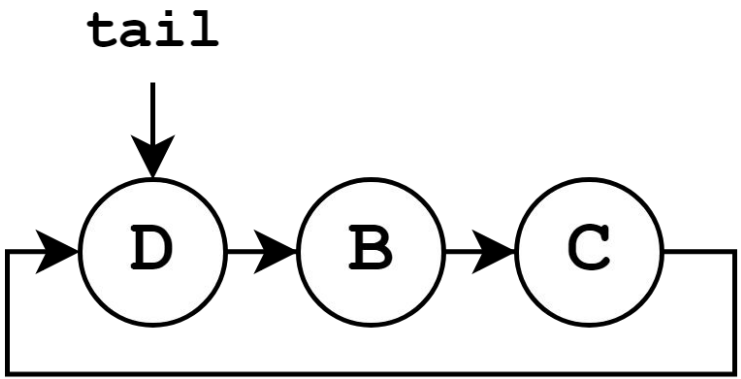
# Queues with Linked Lists

```
queue.offer('A')
queue.offer('B')
queue.offer('C')
queue.offer('D')
```



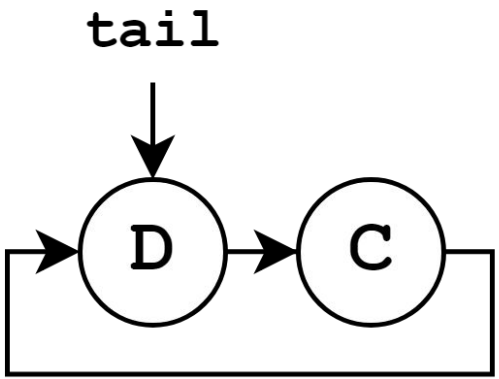
# Queues with Linked Lists

```
queue.offer('A')
queue.offer('B')
queue.offer('C')
queue.offer('D')
queue.poll() //A
```



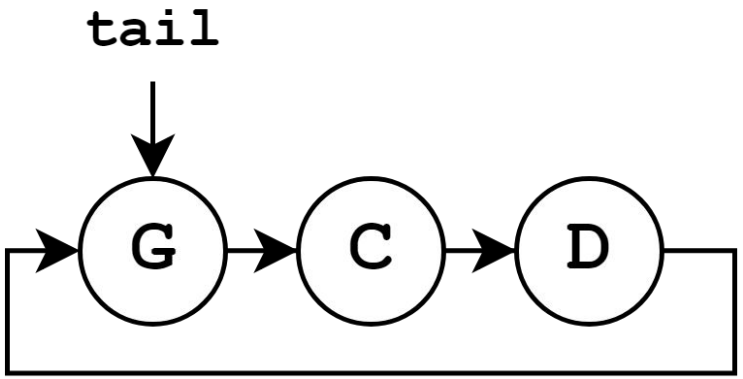
# Queues with Linked Lists

```
queue.offer('A')
queue.offer('B')
queue.offer('C')
queue.offer('D')
queue.poll() //A
queue.poll() //B
```



# Queues with Linked Lists

```
queue.offer('A')
queue.offer('B')
queue.offer('C')
queue.offer('D')
queue.poll() //A
queue.poll() //B
queue.offer('G')
```



# Queues with Linked Lists

**Offer(E data)**

```
n = new Node(data)
n.next = tail.next
tail.next = n
```

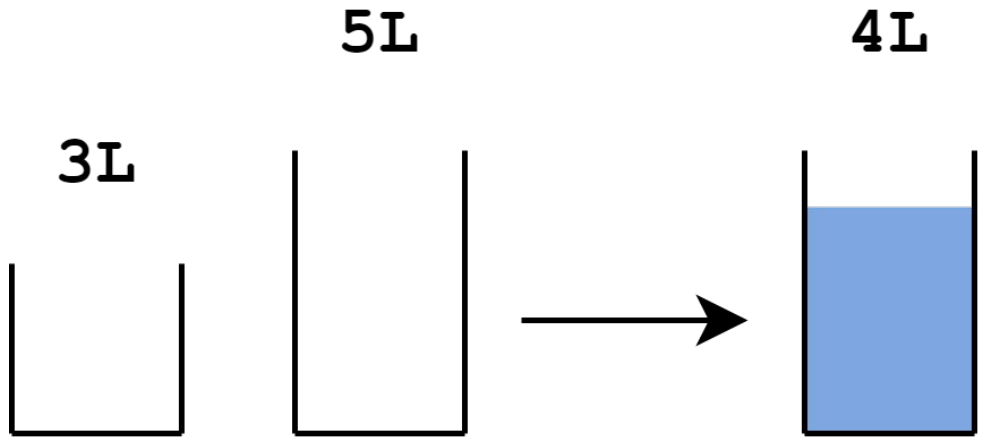
**Poll()**

```
data = tail.next.data
tail.next =
    tail.next.next
return data
```

# Stack/Queue Problems

# Warmup

Given two buckets, a 3-liter and a 5-liter, and an infinite supply of water, describe how you would obtain exactly 4 liters of water.



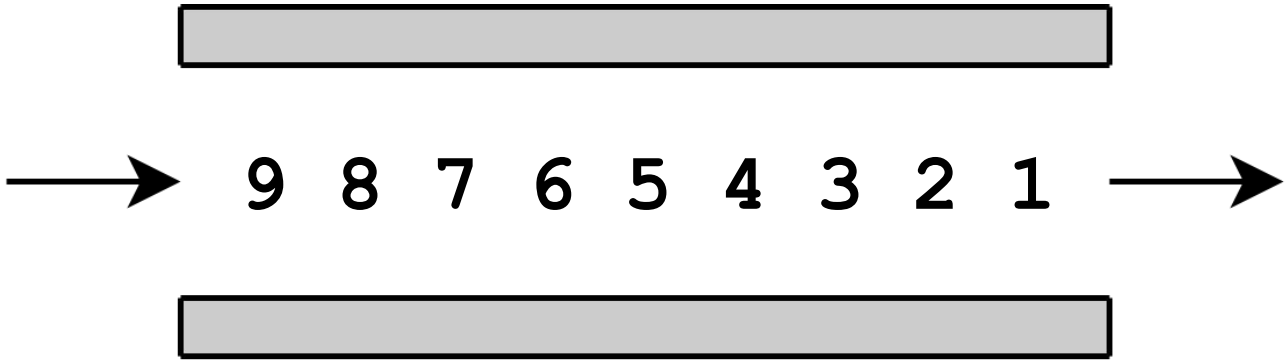
# Problem 1

Reverse a list with a stack

```
[A, B, C, D] -> [D, C, B, A]
```

# Problem 2

Remove every other element from a queue



# Problem 3

Use a stack to check if a string has an even number of brackets

Even:

[ [ { } ] ( ) ]

Not Even:

[ [ ( { ) } ] ]

# Problem 4 (difficult)

Implement a queue using only two stacks