QUIZ QUEUES JAVASCRIPT

Add code that checks if the size of a queue is below its maximum size.

```
hasRoom() {
  return [this.size < this.maxSize] ;
}
```

👏 You got it!

Finish building out this `.enqueue()` method.

```
enqueue( [data] ) {
  this.queue. [addToTail] (data);
  this.size [++] ;
}
```

👏 You got it!

What condition would occur if we ran this code?

```
const colleges = new Queue(3);

colleges.enqueue('Chiptune College of Music');
colleges.enqueue('Breadboard School of Engineering');
colleges.enqueue('8-Bit Art College');

for (let i = 0; i < 4; i++) {
  colleges.dequeue();
}
```

Underflow

👏 That's right! After dequeueing all 3 of the nodes stored in the queue, `universities`, our the loop would attempt to dequeue an already empty queue.

Overflow

Which term best describes the type of queue we could create with the class shown below?

```
class Queue {
  constructor() {
    this.queue = new LinkedList();
    this.size = 0;
  }
}
```

Unbounded queue

Excellent! Yes, this class would create a queue without a maximum size property. We can add as many nodes as we want to an instance of this class.

Bounded queue

If we add a final statement calling `.dequeue()` on `musicPlaylist` what value would be returned?

```
const musicPlaylist = new Queue(5);

musicPlaylist.enqueue('Song 1');
musicPlaylist.enqueue('Song 2');
musicPlaylist.dequeue();
musicPlaylist.enqueue('Song 3');
```

`'Song 1'`

`'Song 2'`

Correct! The queue originally contained nodes in this order: `'Song 1'`, `'Song 2'`, `'Song 3'`. After dequeuing once we have a queue of: `'Song 2'`, `'Song 3'`. If we dequeue again, the value returned would be... Show more

`'Song 3'`

Finish the constructor method of class `Queue` by filling in each of the instance properties of a queue.

```
class Queue {
  constructor(maxSize = Infinity) {
      this.queue   = new LinkedList();
      this.maxSize   = maxSize;
      this.size   = 0;
  }
}
```

You got it!

After running this code, which of the following statements is true?

```
const muffinsToBeEaten = new Queue();

muffinsToBeEaten.enqueue('blueberry');
muffinsToBeEaten.enqueue('corn');
muffinsToBeEaten.dequeue();
```

The value returned after running this code is 'corn'.

The head node is also the tail node.

Yes! Dequeuing `muffinsToBeEaten` leaves only one node in the queue. If a queue has only one node, that node is both the head and the tail node.

There are two nodes in the queue, one for 'corn' and one for 'blueberry.'

Finish building out this `.dequeue()` method.

```
dequeue() {
  const data = this.queue.   removeHead()   ;
  this.   size--   ;
  return   data   ;
}
```

You got it!

Add code that checks if a queue is empty.

```
isEmpty() {
  return   this.size === 0   ;
}
```

You got it!

What condition would occur if we ran this code?

```
const openParkingSpaces = new Queue(3);

openParkingSpaces.enqueue('Tesla Roadster');
openParkingSpaces.enqueue('Ford Pinto');
openParkingSpaces.enqueue('Citreon DS');
openParkingSpaces.enqueue('Generic White Van');
```

Overflow

This is a bounded queue with a maximum size of 3. Adding a fourth node would not be possible, since the queue is full.

Underflow

Which of the following conditions would allow the `while` loop to enqueue multiple strings stored in `names` AND avoid overflow?

```
const zooCollection = new Queue(3);
const cryptids = ['Bigfoot', 'Loch Ness Monster', 'Yeti', 'Trunko'];

while(   zooCollection.hasRoom()   ) {
  const name = cryptids.pop();
  zooCollection.enqueue(name);
}
```

You got it!

Which of the following real-world situations could best be represented with a queue?

Eggs in a carton

People waiting to see a doctor at the hospital emergency room

That's correct! We can represent the patients as nodes in a queue. The first patient to arrive is the first to be seen by the doctor, (FIFO order).

Putting several bangle bracelets on your wrist, then taking them off