Prepared by Linan Qiu <lq2137@columbia.edu>, adapted from Open Data Structures (opendatastructures.org)

(Bonus) Singly Linked List with Tail

Remember we mentioned that `addLast` and `removeLast` are $O(N)$ because we have to traverse to the very end of the list? Turns out sometimes we do need to use `addLast` (much less so `removeLast` and there's nothing we can do about `removeLast` being $O(N)$ for singly linked lists), and we don't want it to be $O(N)$. Fortunately, there's a trick we can do. We'll just do a sketch of the implementation.

Let's call this class `AwsmerLinkedList`

This involves appending a `tail` to the entire linked list. In other words, the class would look like

```
// AwsmerLinkedList.java

public AwsmerLinkedList<T> implements AwsmList<T>, Iterable<T> {
  private AwsmNode<T> head;
  private AwsmNode<T> tail;
  private int tail;

  public AwsmLinkedList() {
    tail = new AwsmNode<>(null, null);
    head = new AwsmNode<>(null, tail); // AHA!
  }
}
```

So now the linked list looks like this:

```
[head][tail]
```

Now think of the `tail` node as a phantom 'shell' node. Why? Look at how we modified the `addLast` method:

```
public void addLast(T item) {
  AwsmNode<T> newTail = new AwsmNode<>(null, null);
  tail.data = item;
  tail.next = newTail;
  tail = newTail;
}
```

See the trick there? If you don't, here's a breakdown:

Let's say we want to add `A`. Instead of creating a new car around `A`, we overwrite `A` into the tail node's `data`.

```
we start with this
[head][tail]

overwrite tail's data with A
[head][A]

then create a new tail node
[head][A]     [tail]

attach the new tail node to A, since we have control over [A] (the previous tail)'s next
[head][A][tail]
```

Isn't this cool? This gives us $O(1)$ `addLast` which is absolutely crucial in a singly linked list functioning as a queue.