CSCI 132:
Basic Data Structures and Algorithms

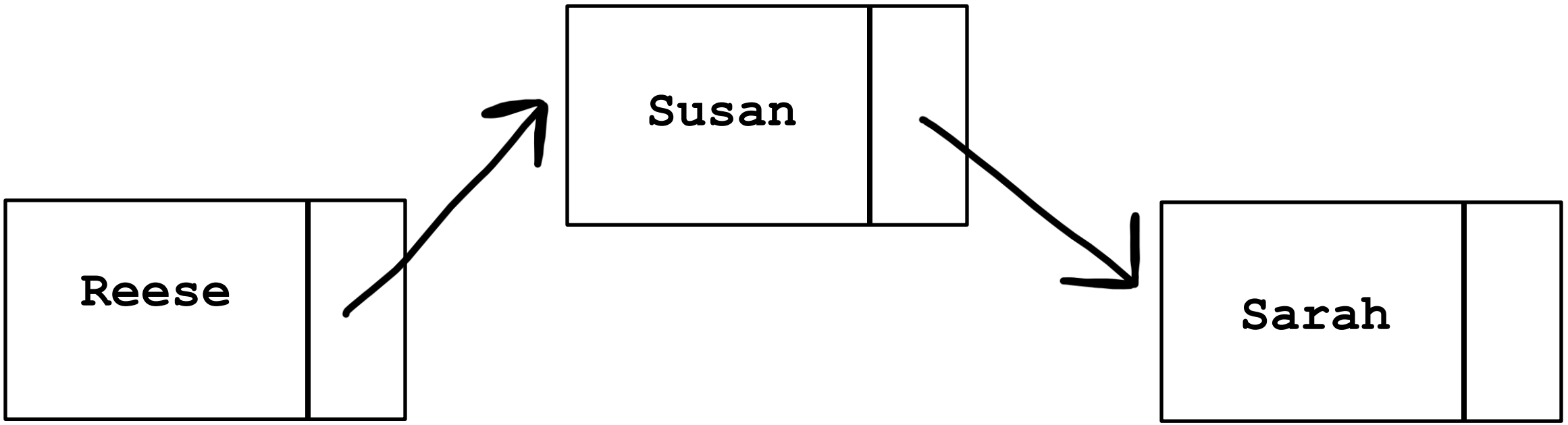
Linked Lists (Part 4)
Circular Linked Lists, Program 2

Reese Pearsall & Iliana Castillon
Fall 2024

Program 2 Posted

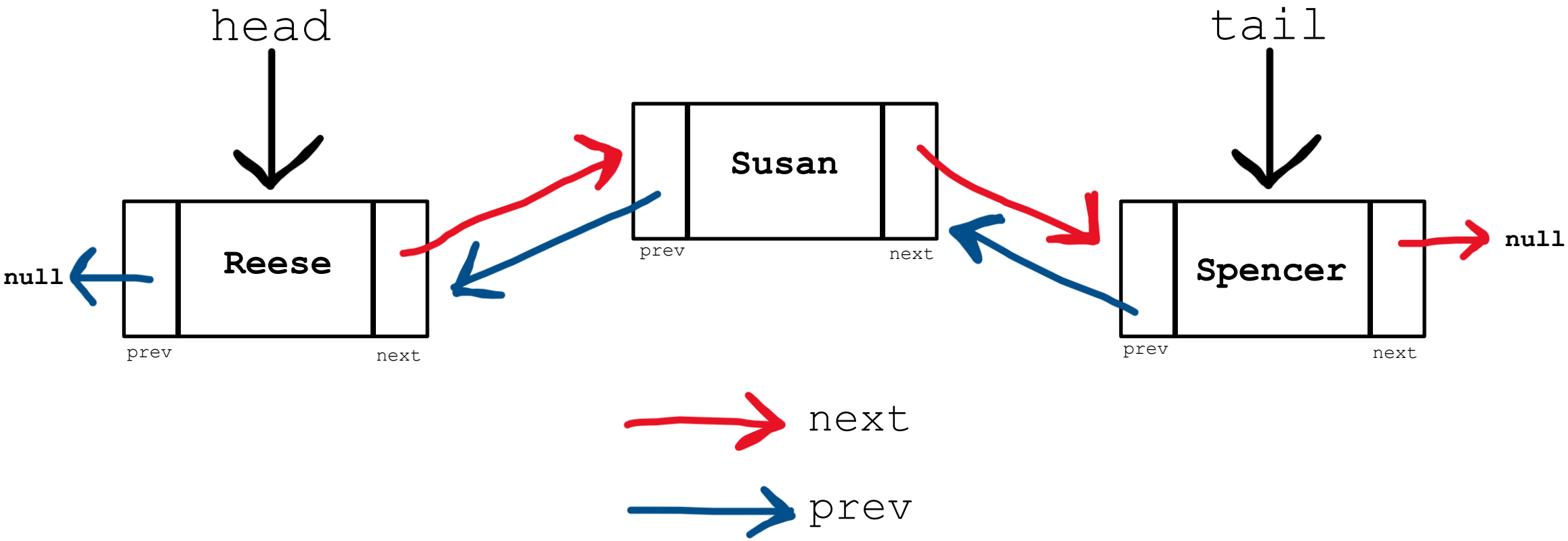
- Due two weeks from now

A **Linked List** is a data structure that consists of a collection of connected nodes

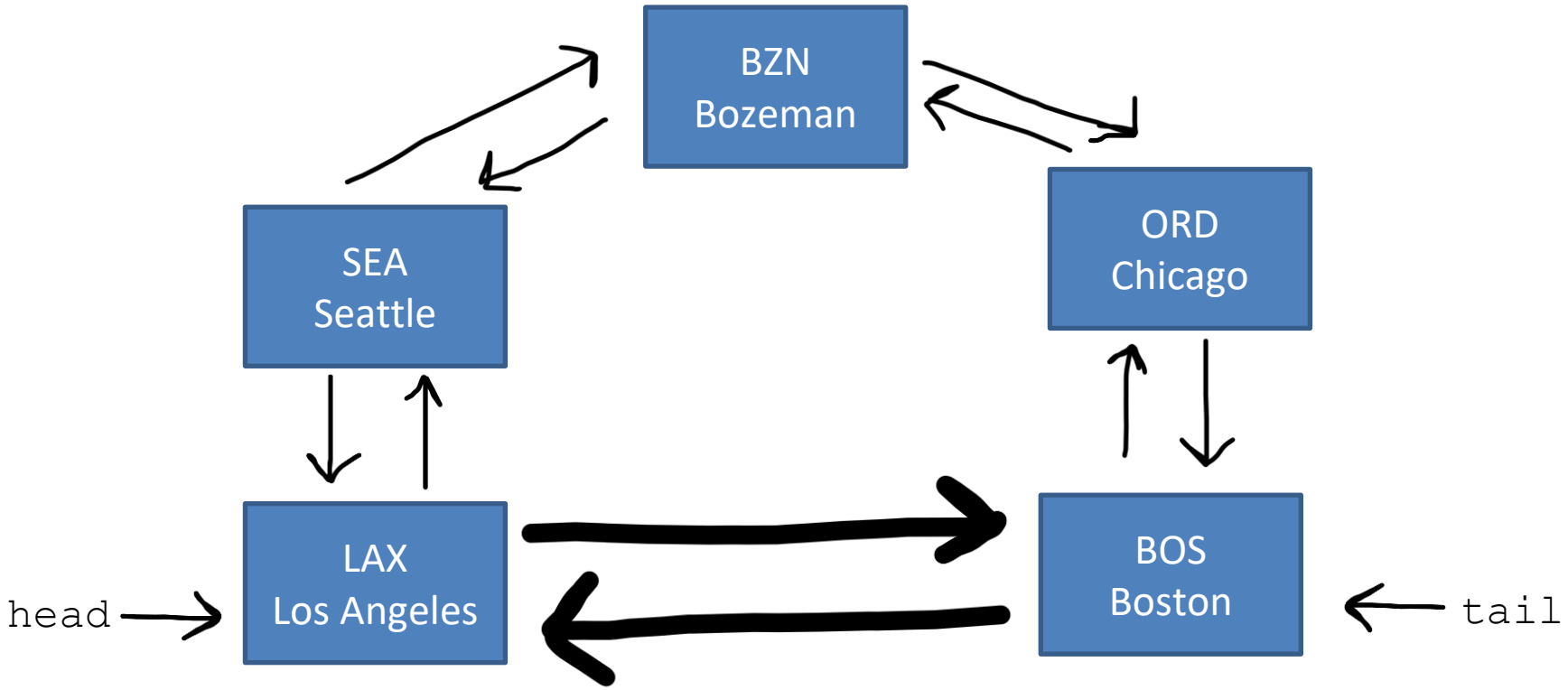


Nodes consists of **data** (String, int, array, etc) and a **pointer to the next node**

A **Doubly Linked List** keeps track of the next node and the previous node

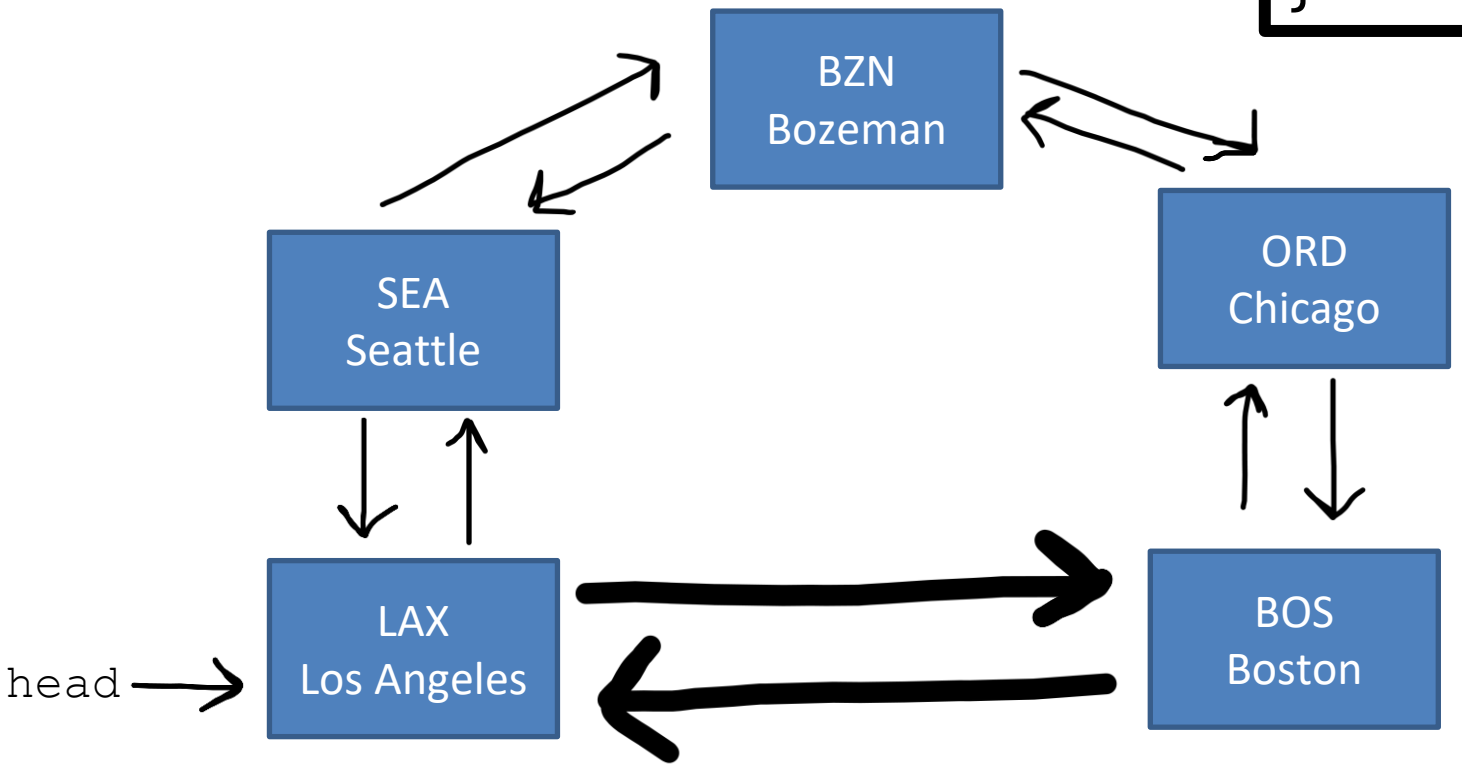


A **Circular Linked List** is a linked list where the first and last node are connected, which creates a circle



Traversing a Circular Linked List

```
public void printList() {
    Node current = this.head;
    while(current != null) {
        current.printNode();
        current = current.getNext();
    }
}
```

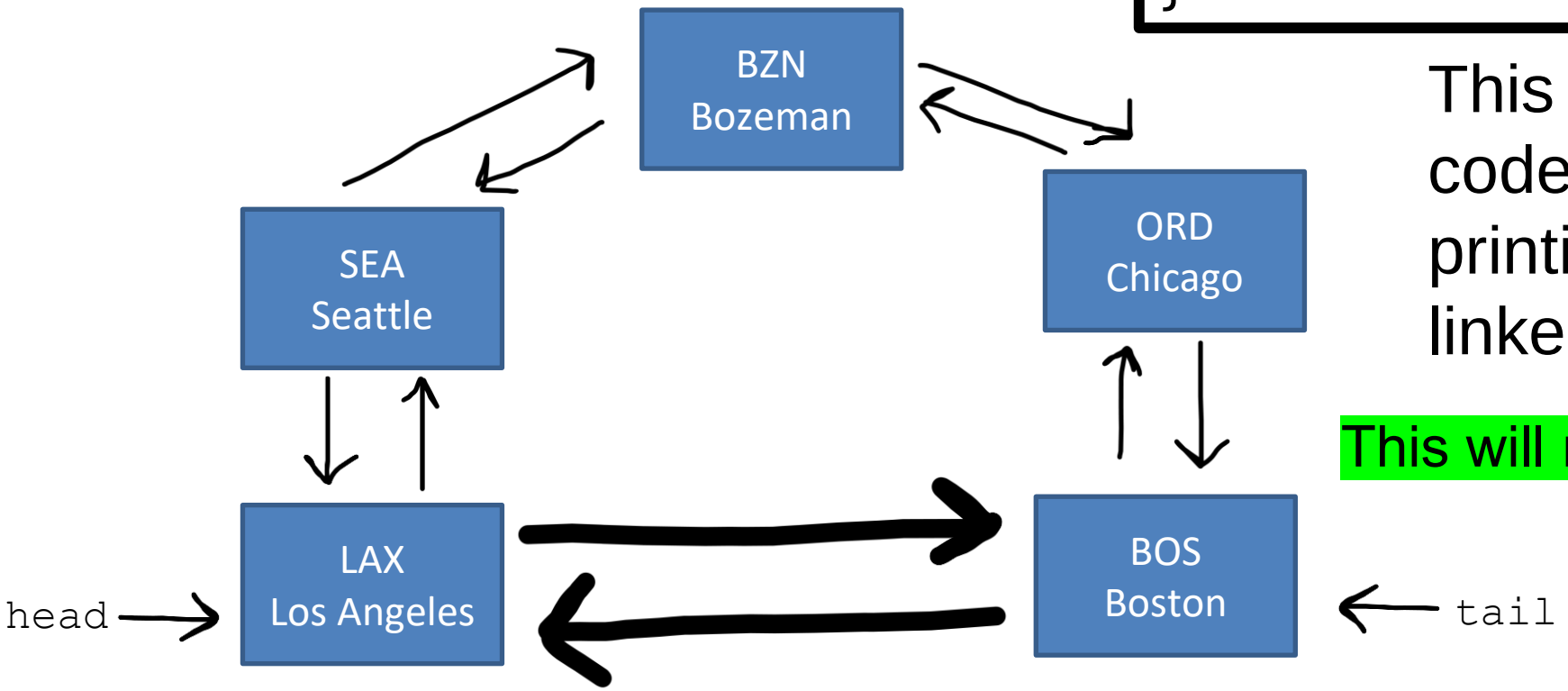


This was our previous code for traversing and printing out nodes in a linked list

This will no longer work because…

Traversing a Circular Linked List

```
public void printList() {
    Node current = this.head;
    while(current != null) {
        current.printNode();
        current = current.getNext();
    }
}
```

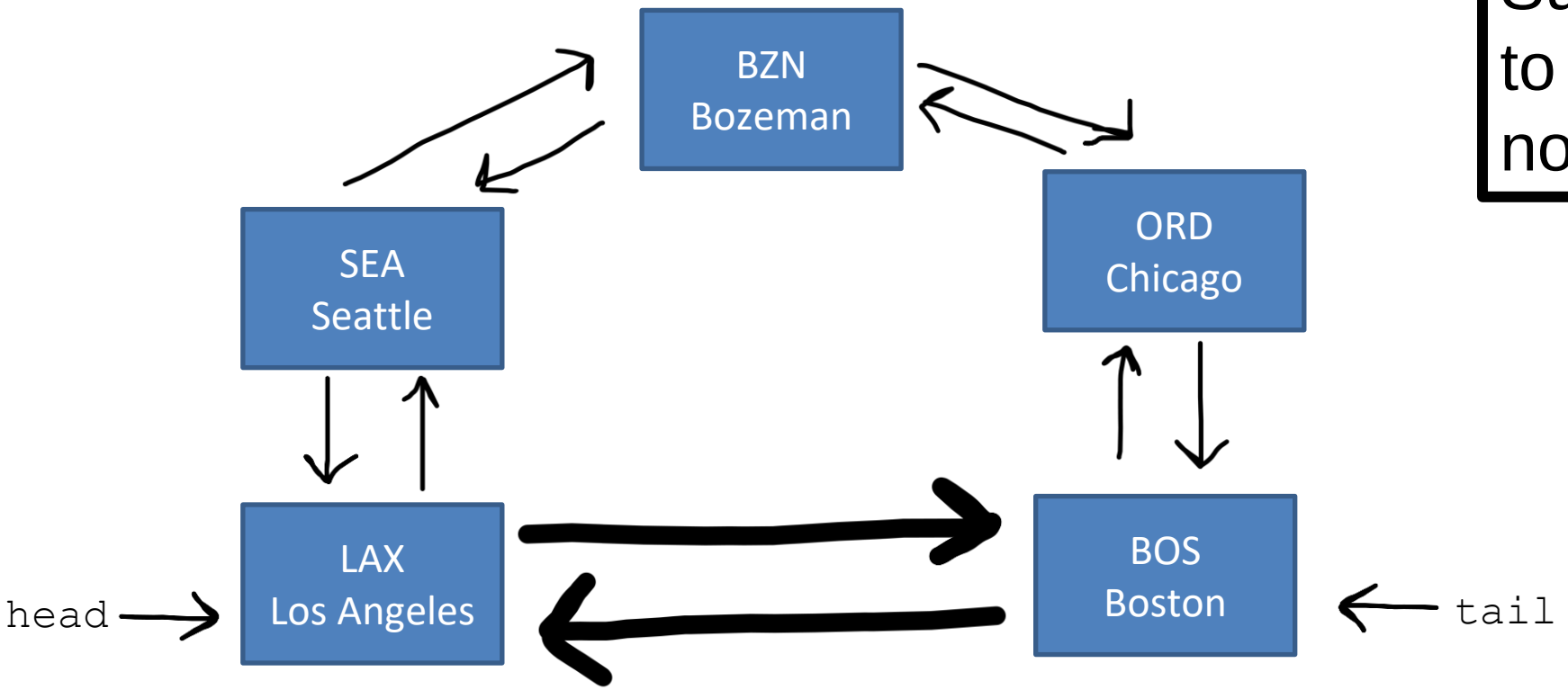


This was our previous code for traversing and printing out nodes in a linked list

This will no longer work because…

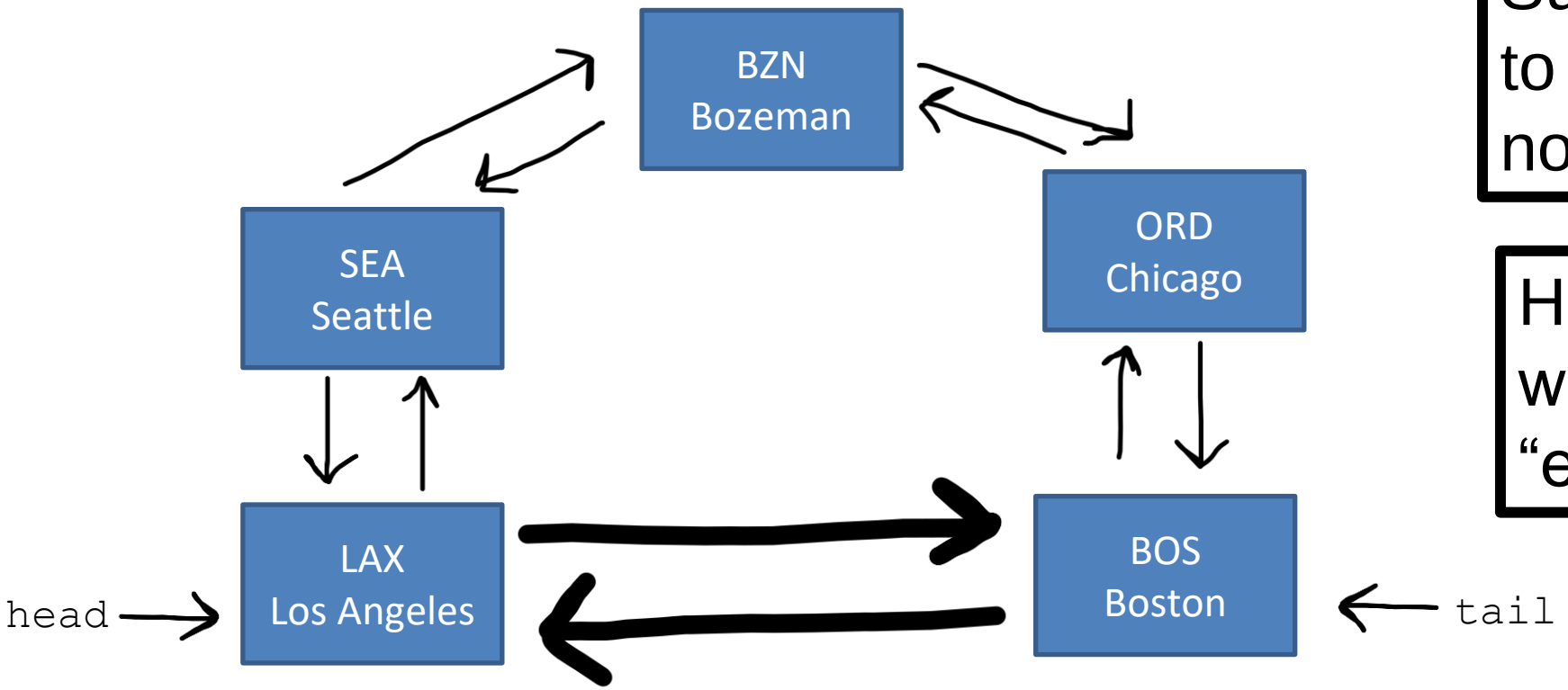
We will never reach `null`

Traversing a Circular Linked List



Suppose our goal is to print out each node only once

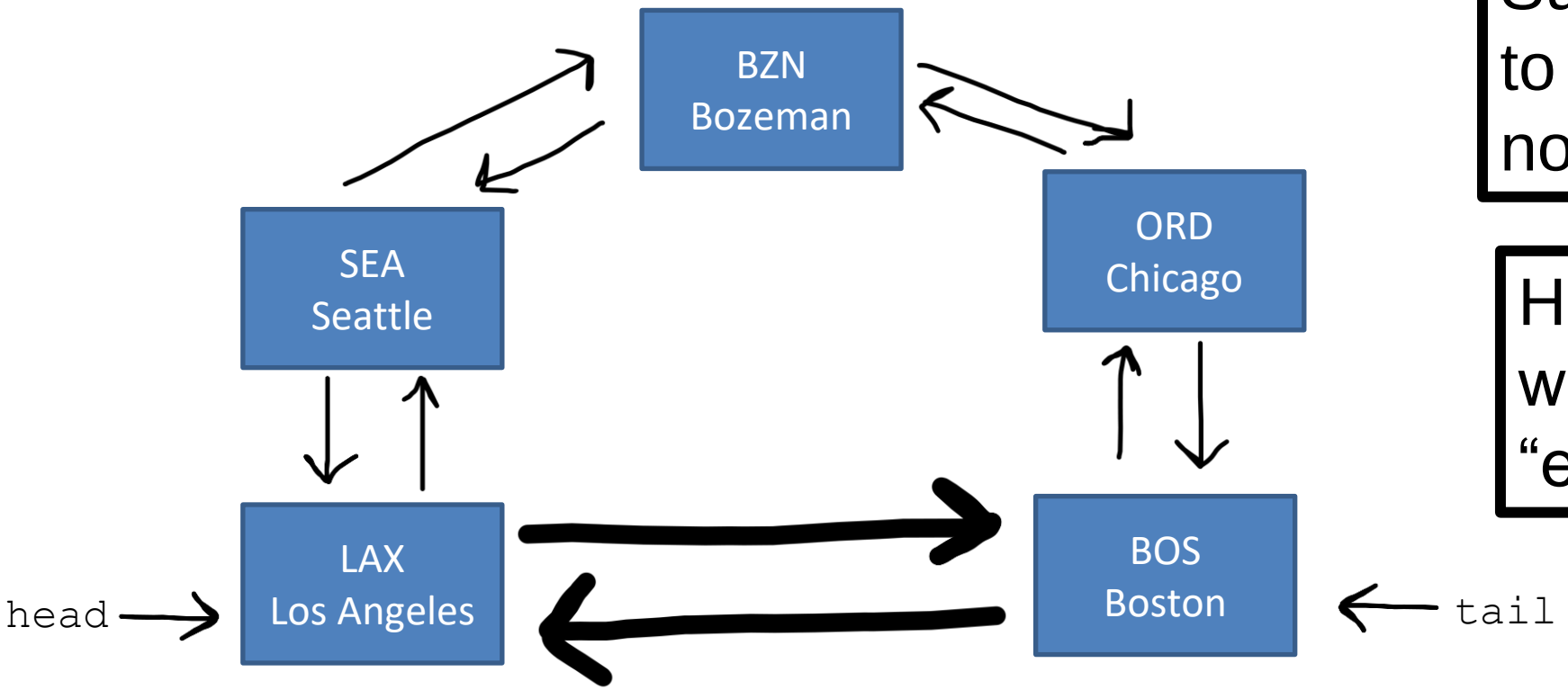
Traversing a Circular Linked List



Suppose our goal is to print out each node only once

How do we know that we've reached the "end" of the CLL ?

Traversing a Circular Linked List

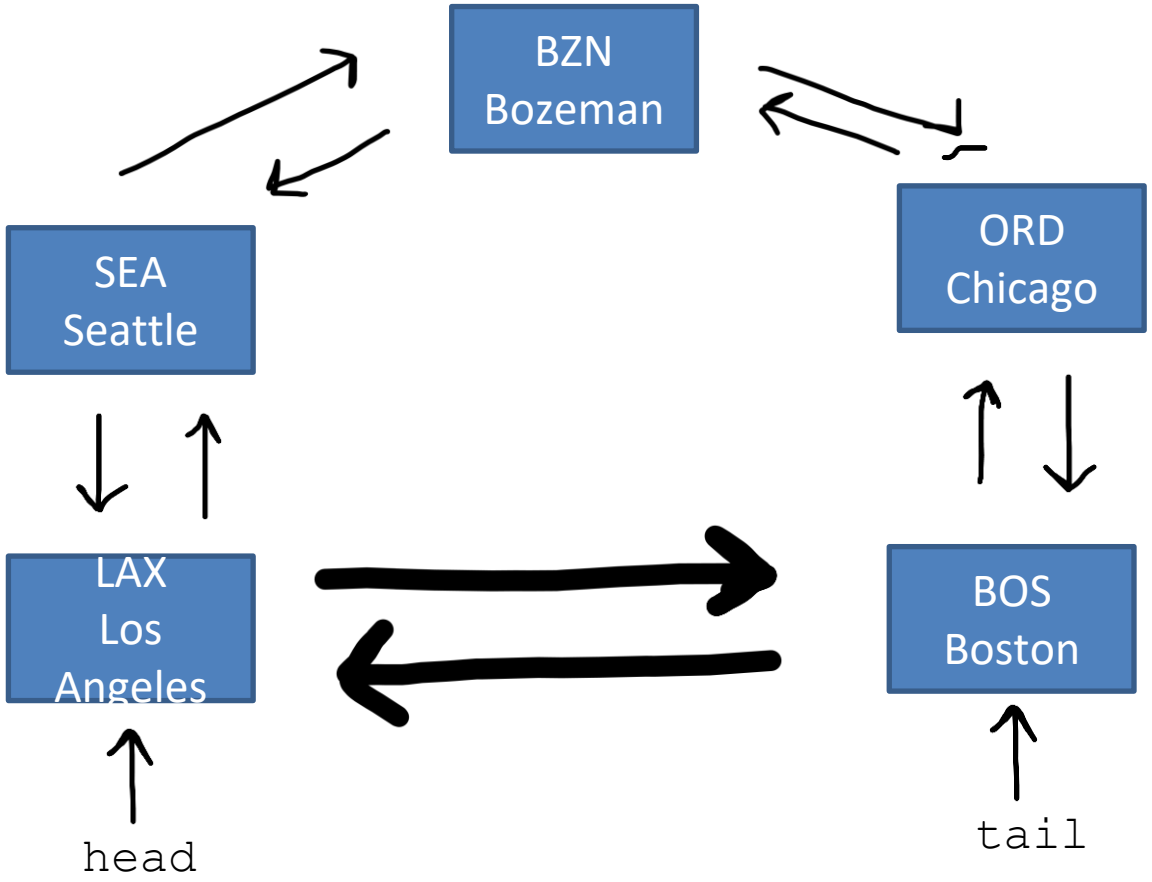


Suppose our goal is to print out each node only once

How do we know that we've reached the "end" of the CLL ?

If we start from the `head`, we should stop looping once we reach the `head` again

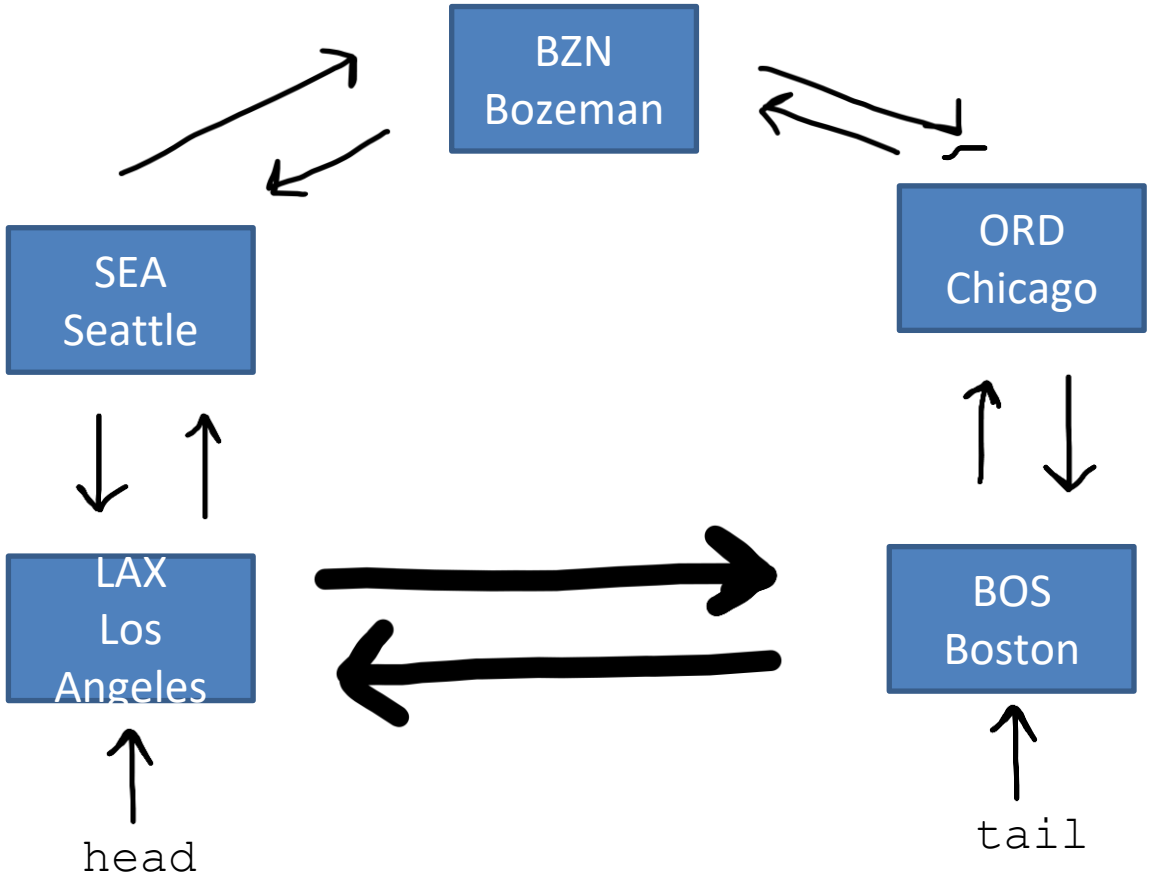
Traversing a Circular Linked List



If we start from the `head`, we should stop looping once we reach the `head` again

```
public void printLinkedList() {
    Node current = this.head.getNext();
    while(current != this.head) {
        current.printNode();
        current = current.getNext();
    }
}
```

Traversing a Circular Linked List

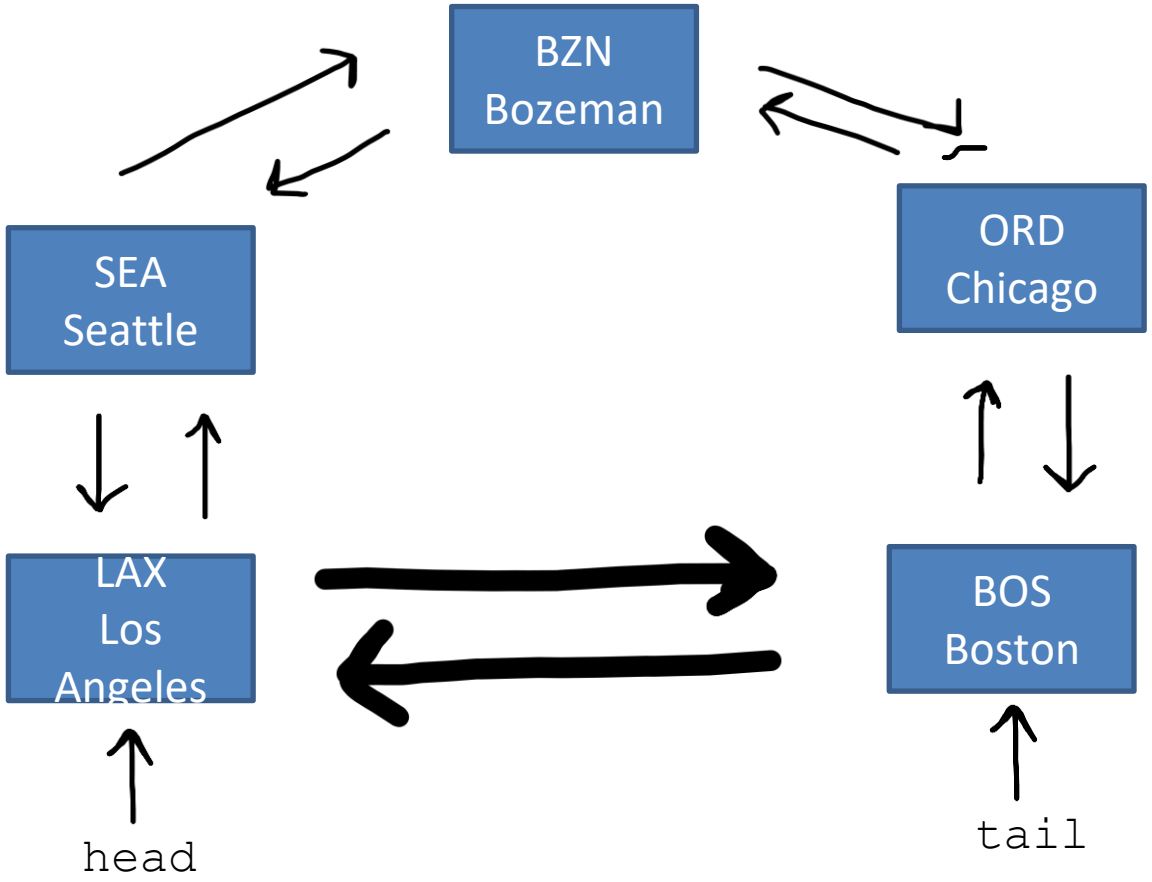


If we start from the `head`, we should stop looping once we reach the `head` again

```
public void printLinkedList() {
    Node current = this.head.getNext();
    while(current != this.head) {
        current.printNode();
        current = current.getNext();
    }
}
```

This **won't** work because…

Traversing a Circular Linked List

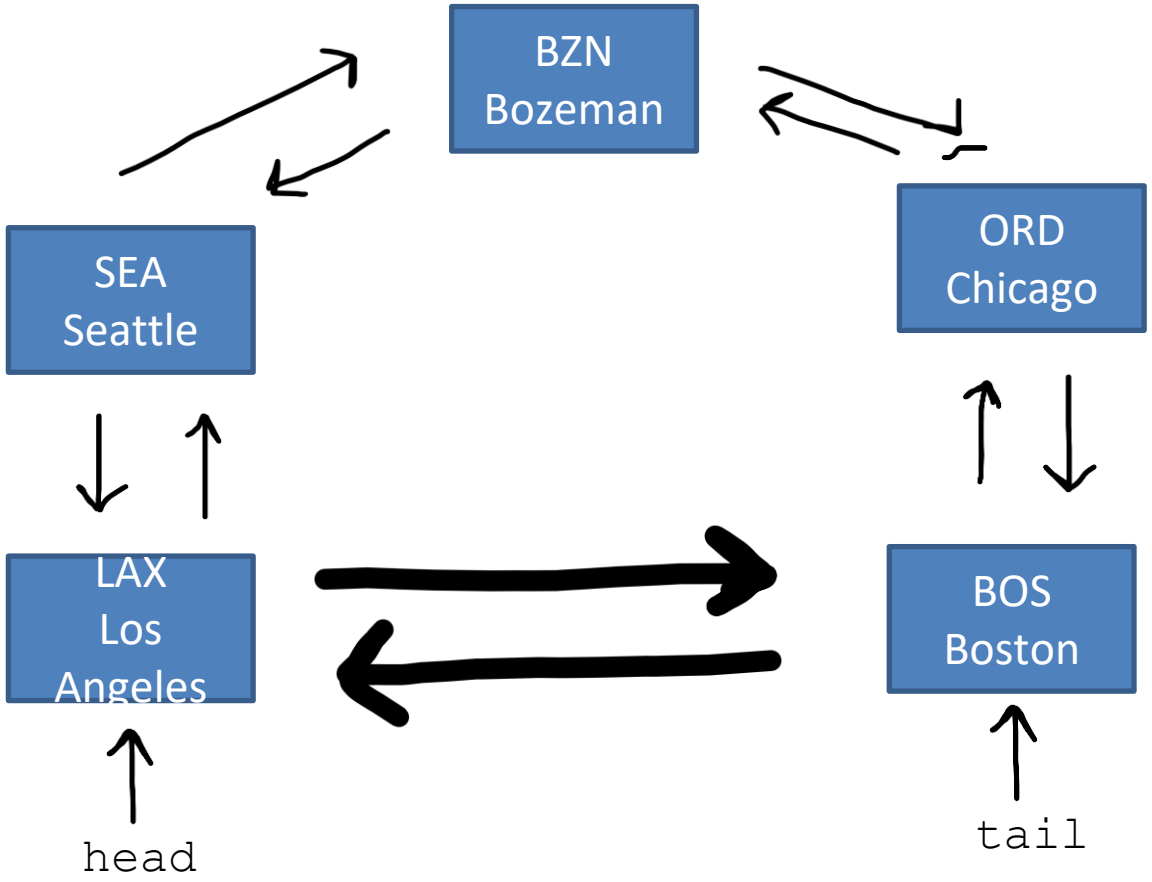


If we start from the `head`, we should stop looping once we reach the `head` again

```
public void printLinkedList() {
    Node current = this.head.getNext();
    while(current != this.head) {
        current.printNode();
        current = current.getNext();
    }
}
```

This **won't** work because… The `head` node will never be printed out

Traversing a Circular Linked List



If we start from the `head`, we should stop looping once we reach the `head` again

```
public void printLinkedList() {
    Node current = this.head;
    do {
        current.printNode();
        current = current.getNext();
    }
    while(current != this.head);
}
```

A **do/while** loop executed the body of the loop, and then checks the looping condition