## Recursion

Recursion occurs when a function is called within its own definition.

Recursive algorithms have three properties:

-

-

-

Why is recursion useful?

-

-

At first glace, it might appear that using recursion to solve a problem is more complicated and difficult than using loops. Later in the semester, we will see that it can be easier to use recursion to solve certain problems.

General Template

In general, our recursive algorithms will look like the following:

```
methodName(parameters) {
  if (base-case) {
        return from recursion
  } else {
        ...
        return methodName(updated parameters)
  }
}
```

## Examples

### Sum (array):

```
public static int sum(int[] arr) {

}

/*
 * Purpose: get the sum of all values from i onward
 *          in the given array
 * Parameters: int[] arr - the array
 *             int i – current index
 * Returns: int - the sum
 * Preconditions: i >= 0
 */
private static int sumRec(int[] arr, int i) {
     if (i ==                 ) {
        return 0;
     } else {
        return           + sumArrayRec(arr, i+1);
     }
}
```

Visually:

| arr: | 7 | 2 | 5 | 4 |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |

## Sum (linked-list):

How would we sum all of the elements within a linked list class?
Things to think about:

- how do we visit all of the elements?
- when is the work finished?
- what do we do at each element?

```
public int sumValues() {

}

private int sumRec(Node cur) {
     if (cur == null) {
        return 0;
     } else {
        return                    + sumArrayRec(cur.next);
     }
}
```

## Accumulators

An accumulator is another argument passed to a recursive algorithm. In the lecture video and slides, we used an accumulator to keep track of the element in the list that comes *before* the current element.

A similar problem may be to determine if a certain value, say *x*, occurs in the list three times in a row. Let's start with two base cases:

```
boolean threeInARow(Node cur, int toFind) {
  if (...) {
        return true
  } else if (...) {
        return false
  } else {
        ...
        return methodName(updated parameters)
  }
}
```

## Key Takeaway: