Important Steps

1. Precise Specification
   - What does the method do?
   - What are the preconditions?
2. Write the base case
   - What is the most basic case?
   - What causes termination of the recursive method?
3. Write the recursive case
   - How do we make progress toward termination?
   - Is your computation correct?

What we’ll do today

- Practice writing recursive specifications and functions
- Given a recursive problem definition
  - Determine a proper specification (note preconditions)
- Given a problem description and specification:
  - Write the recursive base case
  - Write the recursive call
  - Verify that it is correct

Writing Specifications

- Write a specification for a Method that:
  3. Calculate the number of digits in an input integer.
     
     /* = number of digits in integer n. */

- Reduce the positive input integer to a single digit.
  ie. 4728 -> 4+7+2+8 = 21 -> 2+1 = 3
  /* = n reduced to a single digit (by repeatedly summing its digits).
  Precondition: n > 0 */

Problem: Properly add commas to an integer and return the string representation. ie. 5923821 is converted to 5,923,821.

```java
/** String representation of integer with commas added */
public static String addCommas(int n) {
    // Base case
    if (n < 1000)
        return "" + n;
    // Recursive Case
    String number = "" + n;
    return addCommas(n/1000) + "", + number.substring(number.length()-3);
}
```
Problem: Properly add commas to an integer and return the string representation. ie. 5923821 is converted to 5,923,821.

```java
/**
 * = String representation of integer with commas added
 */
public static String addCommas(int n) {
    if (n < 0)
        return "-" + addCommasHelper(-n);
    else
        return addCommasHelper(n);
}
```

Complement of an Integer

```java
/**
 * = the complement of n, formed by replacing each decimal digit of n by 10-n.
 * ie. the result for the integer 93723 is 17387,
 * Precondition: n > 0 and no digit of n is 0 */
public static int complement(int n) {
    // Base Case
    if (n < 10)
        return 10 - n;
    // Recursive Case
    String number = "" + n;
    return complement(n/10) * 10 + (10 - n%10);
}
```

Spring 2009 Prelim 3

```java
/**
 * = number of digits for n.
 */
public static int length(int n) {
    // Base case
    if (n == 0)
        return n;
    // Recursive Case
    return 1 + length(n/10);
}
```

Spring 2008 Prelim 3

```java
/**
 * = n reduced to a single digit (by repeatedly summing its digits).
 * Precondition: n > 0 */
public static int addUp(int n) {
    // Base case
    if (n < 10)
        return n;
    // Recursive Case
    return addUp(n/10 + n%10);
}
```