1 Practice with Recursion

ANSWER KEY

Note: if you use different math formulas you will get different code. This is not necessarily wrong.
For each problem, write a recursive math formula, then write a recursive method with code. Use appropriate comments.

1. This function calculates the number of monsters in your closet after a given number of days. It assumes you started with one monster. Every day, each monster divides itself into two monsters. Your function takes a number of days and returns a number of monsters. Ignore negative numbers.

```
f(0) = 1
f(n) = 2 * f(n-1)

int monsters(int n) {
  if (n == 0) return 1;
  else return s * monsters(n - 1);
}
```

2. This function takes in a string and returns a string with every other character of the input string. If "abcdefghij" went in, "acegi" would come out. If "a" went in, "a" would come out. If "ab" went in, "a" would come out.

```
f(s) = "" if length 0
f(s) = s if length 1
f(s) = s[0] if length 2
f(s) = s[0] + f(s.substring(2,end)) otherwise

String everyOther(String s) {
  if (s.length() == 0) return "";
  else if (s.length() == 1) return s;
  else if (s.length() == 2) return s.charAt(0);
  else return s.charAt(0) + everyOther(s.substring(2,s.length()));
}
```

3. This function takes in an array of student names (Strings). One by one, it asks if the student is present and changes the name to "Missing" if the answer is no. (It reads in a boolean using the nextBoolean() method of the Scanner class to see if the student is present). Because an array is a reference to its data, you do not have to return anything.

```
f(array, index) = stop if index too high
f(array, index) = is here? if not, set missing. f(array, index+1)

void attendance(String[] array, int index) {
   if (index >= array.length) return;
   else {
     Scanner in = new Scanner(System.in);
     System.out.println("Is " + array[index] + " here?");
     boolean here = in.nextBoolean();

     if (!here) array[index] = "Missing";
```

```
}

// OK, so this is a silly example and you probly would write this as a
// but it is possible to use recursion.
```