Chapter 12 Handout – Recursion

Example 1

```
public void drawLine(int n)
{
    if (n == 0)
        System.out.println("That's all, folks!");
    else
    {
        for (int i = 1; i <= n; i++)
            System.out.print("*");
        System.out.prihtln();
        drawLine(n - 1);
    }
}
```

The method call `drawLine(3)` produces this output:

```
***
**
*
That's all, folks!
```

Example 2

```
//Illustrates infinite recursion.
public void catastrophe(int n)
{
    System.out.println(n);
    catastrophe(n);
}
```

Try running the case `catastrophe(1)` if you have lots of time to waste!

Note: A recursive method must have a base case!

Example 3

The general recursive definition for $n!$ is

$$n! = \begin{cases} 1 & n = 0 \\ n(n-1)! & n > 0 \end{cases}$$

```
 * Precondition:  n >= 0.
 * Postcondition: returns n! */
public static int factorial(int n)
{
    if (n == 0)      //base case
        return 1;
    else
        return n * factorial(n - 1);
}
```

Example 4

Recall the Fibonacci sequence 1, 1, 2, 3, 5, 8, 13, ... The nth Fibonacci number equals the sum of the previous two numbers if $n \geq 3$. Recursively,

$$\text{Fib}(n) = \begin{cases} 1, & n = 1, 2 \\ \text{Fib}(n-1) + \text{Fib}(n-2), & n \geq 3 \end{cases}$$

```
/* Precondition:  n >= 1.
 * Postcondition: Returns the nth Fibonacci number. */
public static int fib(int n)
{
    if (n == 1 || n == 2)
        return 1;
    else
        return fib(n - 1) + fib(n - 2);
}
```

Example 5

Write a recursive method revDigs that outputs its integer parameter with the digits reversed. For example,

```
revDigs(147)   outputs  741
revDigs(4)     outputs  4
```

```
public static void revDigs(int n)
{
    if (n % 10 == 0){
        System.out.println();
    }
    else {
        System.out.print(n % 10);
        revDigs(n / 10);
    }
}
```

Example 6

- Consider the following recursive method:

```
public static int mystery(int n) {
    if (n < 10) {
        return (10 * n) + n;
    } else {
        int a = mystery(n / 10);
        int b = mystery(n % 10);
        return (100 * a) + b;
    }
}
```

  - What is the result of the following call?

```
mystery(348)
```

Example 7

- Write a recursive method `pow` accepts an integer base and exponent and returns the base raised to that exponent.
  - Example: `pow(3, 4)` returns 81

  - Solve the problem recursively and without using loops.

Example 8

- Write a recursive method `printBinary` that accepts an integer and prints that number's representation in binary (base 2).
  - Example: `printBinary(7)` prints 111
  - Example: `printBinary(12)` prints 1100
  - Example: `printBinary(42)` prints 101010

place	10	1
value	4	2

32	16	8	4	2	1
1	0	1	0	1	0

  - Write the method recursively and without using any loops.

```
// Prints the given integer's binary representation.
// Precondition: n >= 0
public static void printBinary(int n) {
    if (n == 0) {
        // base case;
        System.out.println();
    } else {
        // recursive case; break number apart
        printBinary(n / 2);
        System.out.println(n % 2);
    }
}
```

Example 9

- Write a recursive method `isPalindrome` accepts a `String` and returns `true` if it reads the same forwards as backwards.

```
- isPalindrome("madam")                          → true
- isPalindrome("racecar")                        → true
- isPalindrome("step on no pets")                → true
- isPalindrome("able was I ere I saw elba")      → true
- isPalindrome("Java")                           → false
- isPalindrome("rotater")                        → false
- isPalindrome("byebye")                         → false
- isPalindrome("notion")                         → false
```

```
// Returns true if the given string reads the same
// forwards as backwards.
// Trivially true for empty or 1-letter strings.
public static boolean isPalindrome(String s) {
    if (s.length() < 2) {
        return true;    // base case
    } else {
        return s.charAt(0) == s.charAt(s.length() - 1)
            && isPalindrome(s.substring(1, s.length() - 1));
    }
}
```