Recursion

- *Recursion* occurs when a method **calls itself**

Sample Recursive Demo:

```
/**
 * A recursive method is implemented.
 */

public class RecursiveDemo {

    public static void showRecursion(int num) {
        System.out.println("Entering method. num = " + num);
        if (num > 1) {
            showRecursion(num - 1); //call to itself!

        }
        System.out.println("Leaving method. num = "  + num);
    }

    public static void main(String[] args) {
        showRecursion(2);  //initial call
    }
}
```

Output:

```
Entering method. num = 2
Entering method. num = 1
Leaving method. num = 1
Leaving method. num = 2
```

- Use recursion whenever a problem can be solved by solving one or more smaller versions of the same problem, and combining results
- The recursive calls solve the smaller problems

Example: Raising a Number to a Power

Ex) 2^4

- Start by rewriting $2^4 = 2 * 2^3$
- Now $2^3 = 2 * 2^2$
- Now $2^2 = 2 * 2^1$
- Now $2^1 = 2 * 2^0$
- Now $2^0 = 1$
- **The final answer is $2 * 2 * 2 * 2 * 1 = 16$**

$$
\begin{aligned}
&2^4 \\
&= 2 * 2^3 \\
&= 2 * 2 * 2^2 \\
&= 2 * 2 * 2 * 2^1 \\
&= 2 * 2 * 2 * 2 * 2^0 \\
&= 16
\end{aligned}
$$

In each case, the power problem is reduced to a smaller power problem, using this formula:

$$x^n = x * x^{n-1}$$

Infinite Recursion

- However, this solution will **never end**!
- When the formula gets to 2^0, it will continue, with $2^0 = 2 * 2^{-1}$ and **so on forever!**
- To prevent this, a recursive solution must have a *base case* that requires no recursion.
- For this example, the base case will be that when the power is 0, 1 is returned

This Power application includes an intPower() method that implements a recursive solution

```
/**
  * A recursive power method is implemented.
  */

  public class Power {

      /**
       * Returns num to the power power
       * pre: num and power are not 0.
       * post: num to the power power has been returned.
       */
      public static int intPower(int num, int power) {
            int result;
            if (power == 0) {
                  result = 1;     //here is our base case when power is 0
            } else {
                  result = num * intPower(num, power-1); // here is the recursive call
            }
            return(result);
      }


      public static void main(String[] args) {
            int x = intPower(2, 5);
            System.out.println(x);
      }
}
```

Output:

32

Programming Exercise:

Create a RecursionFactorial application that returns the factorial of an integer **using recursion**. The factorial of a number is the product of all positive integers from 1 to the number. For example, $5! = 5 * 4 * 3 * 2 * 1$. Computing 5! Could be thought of as $5 * 4!$ Or more generally $n * (n-1)!$. By definition, 0! is equal to 1.

Submit your source code to the Google Doc "ICS4U – Activity Submission Form"