Recursion
Recursive Thinking

- A recursive definition is one which uses the word or concept being defined in the definition itself
  - Not helpful in English class
  - Very helpful in Programming class
- Recursive Programming is a technique in which a method can call itself
- Recursion can provide elegant solutions to certain kinds of problems
Recursive Definitions

- Consider the following list of numbers:
  
  \[24, 88, 40, 37\]

- A list can be defined recursively

  A LIST is a: number
  or a: number comma LIST

- That is, a LIST is defined to be a single number, or a number followed by a comma followed by a LIST

- The concept of a LIST is used to define itself
Recursive Definitions

- The recursive part of the LIST definition is used several times, ultimately terminating with the non-recursive part:

```
number comma LIST
24 , 88, 40, 37
number comma LIST
  88 , 40, 37
number comma LIST
   40 , 37
    number
     37
```
Infinite Recursion

- All recursive definitions must have a non-recursive part
  - The Base Case
  - The Stopping Condition
- If they don't, there is no way to terminate the recursive path
  - Produces an Infinite Recursion
  - Similar to an infinite loop (with the definition itself causing the infinite “loop”)
Recursive Definitions

- Mathematical formulas often are expressed recursively

- $N!$, for any positive integer $N$, is defined to be the product of all integers between 1 and $N$ inclusive

- This definition can be expressed recursively as:

\[
1! = 1 \\
N! = N \times (N-1)! 
\]

- The concept of the factorial is defined in terms of another factorial until the base case of $1!$ is reached
Recursive Definitions

5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!

120
24
6
2
1
Recursive Programming

- Each call to a recursive method sets up a new execution environment, with new parameters and new local variables.
- As always, when the method execution completes, control returns to the method that invoked it (which may be an earlier invocation of itself).
Recursive Programming

- Consider the problem of computing the sum of all the numbers between 1 and any positive integer $N$, inclusive.

- This problem can be expressed recursively as:

\[
\sum_{i=1}^{N} = N + \sum_{i=1}^{N-1} = N + (N-1) + \sum_{i=1}^{N-2} = \text{etc.}
\]
Recursive Programming

```java
public int sum (int num)
{
    int result;
    if (num == 1)
        result = 1;
    else
        result = num + sum (num - 1);
    return result;
}
```
Recursive Programming

main

sum(3)

result = 6

sum

sum(2)

result = 3

sum

sum(1)

result = 1

sum
Recursion vs. Iteration

- Just because we can use recursion to solve a problem, doesn't mean we should.
- For instance, we usually would not use recursion to solve the sum of 1 to N problem, because the iterative version is easier to understand; in fact, there is a formula which is superior to both recursion and iteration!
- You must be able to determine when recursion is the correct technique to use.
Recursion vs. Iteration

- Every recursive solution has a corresponding iterative solution.
- For example, the sum (or the product) of the numbers between 1 and any positive integer N can be calculated with a for loop.
- Recursion has the overhead of multiple method invocations.
- Nevertheless, recursive solutions often are more simple and elegant than iterative solutions.
Towers of Hanoi

• The *Towers of Hanoi* is a puzzle made up of three vertical pegs and several disks that slide on the pegs.

• The disks are of varying size, initially placed on one peg with the largest disk on the bottom with increasingly smaller disks on top.

• The goal is to move all of the disks from one peg to another according to the following rules:
  • We can move only one disk at a time.
  • We cannot place a larger disk on top of a smaller disk.
  • All disks must be on some peg except for the disk in transit between pegs.
Towers of Hanoi

- To move a stack of $N$ disks from the original peg to the destination peg
  - move the topmost $N - 1$ disks from the original peg to the extra peg
  - move the largest disk from the original peg to the destination peg
  - move the $N-1$ disks from the extra peg to the destination peg
  - The base case occurs when a “stack” consists of only one disk
- This recursive solution is simple and elegant even though the number of move increases exponentially as the number of disks increases
- The iterative solution to the Towers of Hanoi is much more complex