Assignment 6

~Kals
Recursions

**Dilbert / Scott Adams**

- It was theoretically impossible to work this week.
- Everything I needed to do required me to do something else first, until it all looped back on itself like a Möbius strip.
- Maybe you could make a to-do list.
- As if I had a pencil.
Problem 1

- No coding.
- Recursion – Splitting a big problem into smaller ones.
  Eg – factorial of n numbers:
  \[ \text{Factorial}(3) = 1 \times 2 \times 3 \]
  \[ \text{Factorial}(3) = (1 \times 2) \times 3 \]
  \[ = \text{Factorial}(2) \times 3 \]
  \[ = \text{Factorial}(2) = \text{Factorial}(1) \times 1 \]
  \[ = \text{Factorial}(1) = 1 \]

- What is the base case?
  \[ \Rightarrow \text{Factorial}(1) = 1 \]

- What is the recursive case?
  \[ \Rightarrow \text{Factorial}(n) = \text{Factorial}(n-1) \times n \]
Problem 2

• Use the code template – RecursiveGCD.java
• Refer Assignment 3 – Problem 3
• In RecursiveGCD.java you should code this-
  → computeGCD(<int>,<int>) – which returns an <int>
  → computeGCD method should handle the base case:
    Base case: GCD(a,0) = a
  → If given input does not match a base case then
    computedGCD method should recursively call ‘itself’
    using the logic:
    Recursive case: GCD(a,b) = GCD(b,a%b)
• Imp - Ensure that you handle negative numbers also.
Problem 3

- Use the code template – RecursiveChoices.java
- Refer Assignment 4 – Problem 1
- In RecursiveChoices.java you should code this-
  - computeChoices(<int>,<int>) – which returns an <int>
  - the values passed to computeChoices methods are M(total number of items) and N (number of items to choose)
  - computeChoices method should test two base cases:
    - If M = 1 then it should return N
    - If N=M then it should return 1
  - computeChoices method should also have a recursive case:
    \[
    \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \text{ for } 0 < k < n,
    \]

Refer wikipedia - http://en.wikipedia.org/wiki/Combination
Problem 3 (continued)

- Eg. Choices(3,2) = 3! / (3-2)!2! = 3
- The way we solve this problem recursively is as following:
  Choices(3,2) = 2! / (2-1)!1! + 2! / (2-2)!2!
  = 2 + 1
  = 3