 CSc 212 Data Structures

## Programming Project 5 (Chapter 9) - Recursive Thinking

**The Assignment:**

You will implement and test four short recursive functions. With the proper use of recursion, none of these functions should require more than a dozen lines of code.

**Purposes:**

Ensure that you can write and test small recursive functions.

**Before Starting:**

Read all of Chapter 9, especially Sections 9.1 and 9.3.

**Due Date:**

Monday, July 10th, 1:30 pm

**Files that you must write and turn in:**

rec\_fun.cpp: This file should contain the implementations of the four functions described below. You also need to put the functions prototypes in a separate file rec\_fun.h rec\_fun\_test.cpp: write a test program that includes rec\_fun.h and show the output

#### 1. One Binary Number

Write a function with this prototype:

  void binary\_print (ostream& outs, unsigned int n);

The function prints the value of n as a BINARY number to the ostream outs. If n is zero, then a single zero is printed; otherwise no leading zeros are printed in the output. The '\n' character is NOT printed at the end of the output.   
EXAMPLES:

  n=0 Output:0  
  n=4 Output:100  
  n=27 Output:11011

**NOTE: Your recursive implementation must not use any local variables.**

#### 2. Triangle Pattern

  void triangle (ostream& outs, unsigned int m, unsigned int n)  
  // Precondition: m <= n  
  // Postcondition: The function has printed a pattern of 2\*(n-m+1) lines  
  // to the output stream outs. The first line contains m asterisks, the next   
  // line contains m+1 asterisks, and so on up to a line with n asterisks.  
  // Then the pattern is repeated backwards, going n back down to m.  
  /\* Example output:  
     triangle(cout, 3, 5) will print this to cout:  
     \*\*\*  
     \*\*\*\*  
     \*\*\*\*\*  
     \*\*\*\*\*  
     \*\*\*\*  
     \*\*\*  
  \*/

Hint: Only one of the arguments changes in the recursive call. Which one?

#### 3. Power Function (Project 13, page 473)

Rewrite the recursive power function from Fig. 9.10 on page 463, so that the time to compute power(x, n) is log(n):

    double power (double x, int n);   
    // Precondition: If x is zero, then n must be positive.   
    // Postcondition: The value returned is x raised to the power n.

**Hint: use the formula x^2n = x^n \* x^n**

**4.  Repeated and Indented Sentences (Project 1, page 470)**

Write a function that produces output like the following:

This was written by calling number 1.   
  This was written by calling number 2.   
    This was written by calling number 3.   
      This was written by calling number 4.   
      This was ALSO written by calling number 4.   
    This was ALSO written by calling number 3.   
  This was ALSO written by calling number 2.   
This was ALSO written by calling number 1.

In the above example, the recursion stopped when it reached four levels deep, but your function should be capable of continuing any specific level with the following function prototype.

     void indented\_sentences (size\_t m, size\_t n);   
// precondition: m <=n;   
//postcondition: print out the above pattern by calling number from m to n