CSC 2111 **Spring 2013**

Name(s): _____

Lab 9: Recursion

Reference: *C++ Plus Data Structures*
Chapter 7 - "Programming with Recursion"

Objective: To gain a better understanding of how recursion works and programming using recursion.

Files: *none given*

Introduction

A function that calls itself is said to be recursive because it involves a process called *recursion*. In connection with this lab, you should read Sections 7.1 – 7.6 of your book *C++ Plus Data Structures*, which deal with recursion and give several examples. Some problems are naturally recursive. As you will see, recursion has some hidden pitfalls, and so we must be careful when using it.

Example: Recall that the factorial of a positive integer *n* is the product of all of the integers greater than 0 and less than or equal to *n*. For example:

4! = 4 * 3 * 2 * 1 => 24

You could easily write a program to calculate the factorial of any number n. Here is some source code that does exactly that:

```
#include <iostream>
using namespace std;

int main(int argc,char * argv[])
{
   int n = atoi(argv[1]);
   if (n == 0)
   {
      cerr << "error, canot calculate factorial" << endl;
      exit(1);
   }
   unsigned long long fact =1;

   for (int i=n; i>0; i--)
      fact = fact * i;
   cout << n << "! is equal to " << fact << endl;

   return 0;
}
```

Try this program out by typing in the above code, saving it to a file called *fact.cpp*, and compiling it. **Then run the program and fill out the following table**:

Command line	Value
fact 11	
fact 15	

**** Why do you think the variable `fact` is declared as an `unsigned long long` integer type? Answer here:**

Recursion

Now, we want to define the factorial function recursively. Look at the following definition of the factorial function:

```
n! =  1              if  n == 1     // the base case
   =  n*(n-1)!       otherwise      // inductive case
```

Therefore, from the above definition:

1! = 1
2! = 2 * (1!) = 2
3! = 3 * (2!) = 3 * (2 * (1!)) = 6
4! = 4 * (3!) = 4 * (3 * (2!)) = 4* (3 * (2 * (1!))) = 24

**** In the following table, expand 5! and 6! as we did above (i.e., showing the last line):**

Factorial	Expansion
5!	
6!	

Given the recursive definition of factorial above, how do we translate this into C++ code? When writing recursive functions, you must remember to observe three key things:

1. You must code the base cases.
2. You must code the general case. This is where you put the recursive call.
3. You must ensure that the recursive calls make progress toward the base case.