Lab Document and Questions	Lab #13, 12/5/2013
Name: ________________________________	WFU Username: _________________

CSC 111E: Lab #13 – Recursion (LAST LAB!)

Lab Date: Thursday, 12/5/2013 Due Date: Friday, 12/6/2013 @ 5:00pm

Purpose: The purpose of this lab is to have you gain experience in solving problems using recursion and implementing recursion in Visual Basic.

WARNING: You must use recursion to solve these problems, not loops. See the grading rubric.

Program 1: Back to Compound Interest

One of the very first problems we encountered as needing to solve in these CSC 111 labs was compounding interest. We employed loops to be able to support compounding over time. This time around, we will implement the same idea using recursion.

Assume that we are given the following pieces of information about an account:

- Initial balance: The initial amount of money deposited into the account
- Interest rate: The monthly interest rate that the account receives
- Time: The number of months we want to analyze the interest accrual.

Assuming we have the following data:

Initial balance: $15,000 Interest rate: 0.5% a month Time: 18 months

the balance at the end of the 18 months should be: $16,408.93 after adding interest

We can define the compounding interest function recursively as follows:

- If the number of months is 0, then the solution is the balance.
- If number of months remaining is more than 0, then the solution is the result of compounding interest on ((1+rate) * the balance) for one fewer months.

As an example, compoundInterest(15000,0.5,18) can be solved by computing compoundInterest(15075,0.5,17). Note the compoundInterest function takes in three inputs in order (balance, interest rate, number of months) and each time it is called the balance input and the month input are updated. The balance is updated by multiplying by 1.005 (in this example, with a 0.5% rate) and the months are updated by subtracting 1.

Downloading the program *Lab13Program1.zip* from Sakai, complete the *compoundInterest* recursive function so that it employs the recursive problem solving approach defined above. If things are done correctly, you should see the results listed below for each test case.

Initial balance: $15000, Interest rate: 0.5% a month	, Time: 18 months	➔ End Balance: $16,408.93
Initial balance: $20000, Interest rate: 1% a month	, Time: 0 months	➔ End Balance: $20,000.00
Initial balance: $20000, Interest rate: 1% a month	, Time: 12 months	➔ End Balance: $22,536.50

Program 2: Combinations of Items

In a few array labs, we dealt with purchases of sets of products and we were interested in which pairs of products were bought together. One of your company employees is trying to determine the different subsets of items that can be bought and whether or not it would be useful to examine triples of items, quadruples of items, etc. She is interested in determining information such as – if we have 5 items for sale, how many different sets of 3 distinct items can be bought/would we need to keep statistics on?

In mathematics, answering this questions involves computing the number of combinations that can be generated. This type of problem is typically referred to as "N choose K" (in the above example, 5 choose 3). The solution to this problem can be written recursively, using the following definition:

N choose 0 = 1
N choose N = 1
N choose K = (N-1 choose K-1) + (N-1 choose K)

As an example, assuming you have a function "Combinations" which computes N choose K, Combinations(5,3) can be computed by solving Combinations(4,2) + Combinations(4,3).

Downloading the program *Lab13Program2.zip* from Sakai, complete the *Combinations* recursive function so that it employs the recursive definition provided at the top of this page. Note the Combinations function takes two inputs in this order: the N value as an integer and the K value as an integer. If things are done correctly, you should see the results listed below for each test case.

N	**K**	**Combinations(N,K)**	**N**	**K**	**Combinations(N,K)**
5	0	1	6	2	15
5	1	5	6	3	20
5	2	10	6	4	15
5	3	10	6	5	6
5	4	5	6	6	1
5	5	1	8	3	56
6	0	1	10	4	210
6	1	6	15	12	455

[If you are interested, the formula stem from the following rules:
- There is only one way to choose 0 items from a set of N items
- There is only one way to choose all N items from a set of N items
- For K >0 and < N, assume you pick the last listed item for sale. You then have to pick K-1 more items from the remaining N-1 items. Alternatively, if you ignore the last listed item for sale, you have to choose all K items from the remaining N-1 items.

]

Submission
To submit this lab for grading, do the following by Friday, 12/6, at 5:00pm – zip your projects and upload both files into Saki (under Assignments, Lab 13). See below for grading information.

Your grade will be based on the following rubric:

Objective	**Points Available**	**Earned**
Program 1: Uses recursion instead of a loop	-50 if not followed	
Program 2: Uses recursion instead of a loop	-50 if not followed	
Program 1: Correctly implements base case	14	
Program 1: Correctly implements recursive case	28	
Program 2: Correctly implements base case(s)	28	
Program 2: Correctly implements recursive cases	28	
Free points	2	