CSIS 211 – Data Structures
Spring 2017
Lab #1 (Recursion) – Due Date: 11/15/16 at 11:55 pm
Remember to submit ALL files (.cpp,.h etc.) on Moodle – NO EMAILS – no exceptions
You may work in a group of 1-3 for this assignment
EACH group member should submit just to make sure I don't miss any while grading
IMPORTANT! – put EACH group member's name at the TOP of the CODE in comments
If you are a group of 1 put your name only at the top
Make sure and use good design techniques
This will be worth 50 points

For all Programs:
Do NOT forget to put ALL your groups names at the TOP of EACH FILE in comments.
Example:
//LAB 1
//Group: Kristina Shroyer, John Smith, Jane Doe
Do this EVEN if you have a group of 1
Since this is very important

Homework #1 – Recursion and Templates

Note this is a procedural program! Class data types not required. However, I have asked you for function templates in some places. Make sure and use those where asked for – see the examples in the C++ interlude #1 for function templates and the examples in Chapter 2 for recursion.

First write the following recursive functions/function templates:

1. **recursivePower** - Write a recursive function **recursivePower** that computes a^n where a is a real number and n is a non-negative integer. This does NOT have to be a function template. Make sure and write pre and post conditions if they are needed.
2. **maxArray** – This one started in Chapter 2. Implement the **RECURSIVE maxArray** function we talked about in the Chapter 2 (it's also Section 2.4.3 of your book) as a FUNCTION TEMPLATE. That means your solution is required to be a function template. So this should find the maximum of int arrays, double arrays, char arrays (the chars must be all upper case or all lower case) – it won't work on string arrays, so basically any numeric type plus chars. TIP: write it as a non-template first then once that's working change it to a function template.
3. **binarySearch** – Rewrite the binary search algorithm from Chapter 2 as a template (so this one is required to be a template). So it should search int arrays, double arrays, string arrays – basically any type of array where the < and > operators are defined.
4. **reverseDigits** – write a recursive function that takes an integer as an argument and returns that integer with the digits reversed. The function is REQUIRED to take an integer as an argument. You do NOT need and should NOT have a template for this one.
    a. **Hint:** Remember integer division only returns the whole part of the result – so 10/3 = 3 with integer division, it is not 3.333333333.
    b. Remember mod returns the remainder for integer division so 10 % 3 = 1
    c. So a positive number % 10 gives us the ones digit of that number (201 % 10 = 20 remainder 1…1 is the mod and is the ones digit)

            i. So the idea behind the recursive algorithm in this case to print the ones digit of the number and then divide the number by 10 using integer integer division to lose that ones digit
                1. Example 201/10 = 20

5. Test your recursive functions with a main. Main does NOT need to be fancy but make sure and do multiple tests and for the templates test them on multiple data types. So run maxArray on at LEAST two different array types (maybe a double array and a string array) and do the same with binary search. Run at least three total tests on binarySearch and maxArray as well. Run at LEAST three tests on recursivePower and reverseDigits. More than three tests is better. I just don't want nothing on the page in terms of tests but at the same time you don't need a fancy loop or anything either.