Programming Assignment #3
CS 202 Programming Systems

***** Make sure to read the Background Information first!**
It applies to all programming assignments this term***

IMPORTANT:

1. **This program is about operator overloading in inheritance hierarchies.**
2. **DO NOT USE dynamic binding for this assignment.**
3. **Begin by (a) understanding the assignment and (b) designing the inheritance hierarchies.** *This can be done prior to learning about operator overloading!*
4. Operator overloading will be covered in Lectures during Week #6 (both lectures) and Lab #6 will reinforce the concepts.

Remember our goals:

This term, the key idea is to break down the problem outlined (below) into small pieces and assign those responsibilities to individual classes. For each assignment, your job will be to create an OO design and program that shows how Object Oriented constructs could be used to solve the problem. You will want to focus on how to design classes that are well structured, efficient, that work together, and where each class has a specific **"job"**. This time you are adding operator overloading as the new syntax for your solution!

Operator overloading is best implemented when we create new **abstract data types**. So, look back at your first two programs. Did you tie the data structures into the midst of your OO designs? Did the designs really become all about the data structure and not so much about the problem at hand? This time, we want to create our own abstract data types implemented with a full set of operators and have them used by our OO programs. Unlike in CS163, we are also creating the application surrounding the abstract data type, and not just the ADT!

Program #3 - The Abstract Data Type - Background

As I was driving home last week, listening to the radio, the Emergency Broadcast System interrupted the show I was listening to. At first I thought it was just a test ("This is a test of the Emergency Broadcast System"), but no – in fact this time it was real. Immediately I wondered, is there a tornado? Certainly not another snow event! Well, in fact, it was a widespread Comcast outage affecting people with land-lines (phones) telling them how to contact 911 if there was an emergency (they told them to use someone's cell phone!). So I thought about it. Who would have received this message? Only people who had the radio on at that time. Certainly TV wouldn't have worked for those affected by a Comcast outage. Right?

So, if there WAS an emergency, how would you be notified if you primary form of communication was down? What if the cell tower(s) were down in your area – what then? For me, I have a number of different ways people can contact me – but this information is not centralized as part of an emergency contact system.

Program #3 - The Abstract Data Type – Your Assignment
Your mission with program #3, is to create an EMERGENCY BROADCAST SYSTEM application that will build a contract tree providing the top 3 ways each individual can be contacted (home phone, work phone, cell, facebook, twitter, etc.).

Data structure:
You will be creating a Table ADT implementing a **BST**, organized by last name – where each node is/has a LLL of everyone with that same last name, sorted by first name. Make sure to provide full support for data structures operations: insert, search, display, remove, removal_all.

You may substitute a Balanced tree for the BST implementation (If you do not elect to implement a balanced tree, then you will be implementing one with your last program (in Java). Balanced tree implementations do not require the implementation of an individual delete/removal operation. (You still need to deallocate all).

AFTER lectures this week: The Table ADT class that you write will need support operators for: =, +, +=, ==, !=, the relational operators, and the ability to input/output data. As we discuss the operators this week, think about how to apply the operators, make sure to stay within the rules of how the operators are expected to behave. You may find that some operators don't apply at all (and therefore shouldn't be implemented).

REMEMBER, EVERY CLASS that has dynamic memory MUST have a copy constructor. New this week, EVERY CLASS that has dynamic memory MUST also have an assignment operator to perform a deep copy!