Programming Assignment #3
CS 202 Programming Systems

*****This program is about operator overloading *****

Primary Goal for program #3:

The primary goal for program #3 is to experience operator overloading in an object oriented environment. We want to correctly create classes that properly support the copy constructor, destructor, and now the assignment operator when dynamic memory is involved. Remember that copy constructors in a hierarchy require the use of initialization lists to cause the base class copy constructor to be invoked.

Your primary goal with program #3 is to apply the functionality of the operators, the appropriate arguments (operand data types) and returned types (residual values for the operators) to the following problem. Think about how the operators are used and try to mimic the behavior in your own classes. How is the returned type used? Who controls the memory of the residual value? The client program (lvalue) or the operator (rvalue). Make sure to pass (and return) by reference whenever possible!

Remember OOP:

With OOP 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 a real-world 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!

Before you design this program, take a 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? The key to OOP in my opinion is to create the design based on the problem prior to applying the data structure. The data structures are about how we handle the data – not necessarily the key to OOP. For this assignment, your application will be the OOP portion of the assignment. Then, implement the data structures in Abstract Data Types (ADTs) with the full support of operators (via operator overloading).

Program #3 – The Problem

For this assignment, we will create a program that ultimately will become an app. But, before it does, we need to test out the idea. Have you ever wanted to get together with friends but you had a hard time connecting with everyone. Maybe you posted an invitation on facebook to all of your friends to meet at Oaks Park or maybe you wanted to get a group together to battle one of the Legendary pokemon that can't be found in the wild. Whatever the reason, connecting with others can be a challenge without the right software. Your job is to create a "discord like" program that will allow people of a group (of your choosing) to set up meeting times and places.

The Data: People should be able to post a meeting and respond to a posting. A posted meeting should include at least the following (you may add to this):

(a) The name of the meeting (e.g. Eclipse Gathering)
(b) The location of the meeting (e.g., Sister's Creekside City Park)
(c) The day and time
(d) Group of people to receive the notice
(e) Keyword (type of activity – e.g., Eclipse)

Responding to a posting should include (a) intent to attend (yes, no, maybe) and the comments (e.g., remember to bring your Solar Shades!). To make this useful, you will need to use an external data file to keep track of the meetings that have been posted and/or responded to.

Making it OO: We will use this concept to develop a OO application to Post meetings and respond to posted meeting. Classes to consider having may include (a) Contact (e.g., a friend's information), (b) Group of Contacts, (c) Meeting, and (d) Calendar of Meetings. You will want to develop at least five classes, as normal, to design the program, using single inheritance as appropriate. Make sure to build an inheritance hierarchy. Break down the common elements and push those up to the base class! Then, have the derived classes handle the differences. Remember to avoid getters as much as possible with these classes – instead implement the functions that actually work ON the data IN the classes that own the data!! This is where you will get the most benefit of OOP.

Searching and the Data structure: For this program to be useful, we need to provide the current set of meetings that are going on – for quick access; we would never use it if we had to scroll through 100's of items. In this case, we will be working with a type of balanced tree based on the type of activity (Keyword) (e.g., find everything related to Pokemon Go)

Each node in the list is to contain a linear linked list of all of the data that is based on that keyword. So all of the events that are about the Eclipse will be in one LLL for a node in the tree. All events for Pokemon Go are in another node's LLL.

Operator Overloading: The key part of this assignment is implement a complete set of operators. The operators to support must include: =, +, +=, ==, !=, the relational operators, and the ability to input/output data. I imagine the [] would be useful as well where the keyword could be provided to the subscript operator for quick retrieval. Remember if you implement the + operator you should implement the +=. But, what about the equality operator? As you decide 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). And, don't forget your copy constructor!

Yes, you CAN now write your own STRING data type, but it can't be the only place you use operator overloading. In fact, if you implement a string class, the operators for that class "do not count" for this assignment.

Questions to ask…about operator overloading

When using operator overloading, remember to ask yourself the following questions:

a) What should be the residual value (and what data type is this)?
   - *Never have a void returning operator!*

b) Should the result be a modifiable lvalue or an rvalue?
   - Lvalues are returned by reference, most rvalues are returned by value.

c) What are the data types of the operator's operands?
   - If the operand isn't changed, then make it a const
   - Remember to pass as a constant reference whenever possible.

d) Is the first operand always an object of class?
   - If so, then it could be a member function.

e) Can the operator be used with constant operands?
   - If the first operand can be a constant, and IF it is a member function, then it should be a constant member function.