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. Every class that manages dynamic memory must have a copy constructor, destructor, **and assignment operator**.

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.

Program #3 – The Problem

The Winter Olympics have started! Did you watch the opening ceremonies? Or, is there a particular sport you are interested in watching. The trouble is that the network is splicing different sports together – where we get to see parts of what we might be interested in (during prime time) rather than selecting just the sport of interest. We have decided to fix this by creating an application that will help the user find out when a particular sport (e.g., down hill skiing) will be taking place and even when we expect a particular athlete to be competing.

The Data: Of course, your job IS NOT to write the actual application. That is something that we would do in industry. Instead, we want to "test market" the idea. This means enough of the framework should be put together that we can use the program but the data does not all need to be realistic yet! In fact, the data needs to come from an **external data file as well as from the user**. The data should include at least the following (you may add to this): (a) Name of the sport (e.g., Figure Skating), (b) Details about the event (e.g., Ladies Single Short Program, Pairs Free Skating), (c) The day and time it airs (e.g., 11:30-12:30am ET), (d) The network (or channel) (e.g., NBC). For each sport you will also need information about the athletes. This should include at least (a) Full Name (e.g., Breezy Johnson), (b) Their event (e.g., Alpine Skiing), and (c) Medals received. You will want to keep some **historical data** as well so we know how well the athlete did previously (such as did they win a medal in a previous Olympics?).

Making it OO: We will use this concept to develop an OO application to help find sports and/or athletes to watch compete. You will want to develop at least five classes, as normal. Make sure to use single inheritance in your design. To break down the problem, think about what is similar or different about different kinds of sports (e.g., team sports versus individual events). Think about how history can fit in with this. Use inheritance, break the common elements of the major types of sports 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: Our data structure will be a TREE of sports. For each "node", we should have a LLL of athletes that compete in that sporting event. The athlete's history of metals should also be supported by a dynamic data structure, of your choice. You have an option with this – the TREE can either be implemented as a Binary Search Tree (BST) or it can be implemented as a balanced

tree. The allowed balanced trees are: 2-3, 2-3-4, or red-black. Or, you can wait until program #5 to implement a balanced tree (in Java).

Operator Overloading: The key part of this assignment is to implement a complete set of operators. The operators to support must include: =, +, +=, ==, !=, the relational/equality operators, and the ability to input/output data. I imagine the [] would be useful as well for searching for a match. As you decide how to apply the operators, make sure to stay within the rules of how the operators are expected to behave.

All of the aforementioned operators NEED to be implemented. But, they do not need to be implemented in the same class. OPERATORS DO NOT COUNT if they are only implemented for a 'string-like' class. Although you MAY write your own STRING data type, it can't be the only place you use operator overloading (since the code is in topic #6!).

Of course, the = operator needs to be implemented in all classes that manage dynamic memory – and don't forget your copy constructors!!!!

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.