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

So many games and software applications have us develop avatars to represent ourselves. Whether it is a game like Halo or a word game like Words with Friends, we have the opportunity to represent ourselves using an avatar.

The Data: Your job for program #3 is to create an Avatar data type that allows users to create an avatar with a variety of different characteristics. Each avatar will need (a) personal characteristics, (b) chest of tools and/or badges won, and (c) history or backstory. The personal characteristics would include information such as their name, nickname, contact information, website, phone number (or other form of contact). Personal characteristics could include visual characteristics such as a graphic, whether or not you wear glasses or a hat. The tool chest might include what badges you have won, the months/years you have played, the number of points (if appropriate for the game), and so on. Please have fun with these lists and come up with some of your own characteristics and tools that are common with the games or applications that you use.

Making it OO: We will use this concept to develop an OO application to help a user develop their avatar. 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 characteristics. Use inheritance to push up the common elements 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 characteristics and tools that are part of the user's avatar. 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 2-3 balanced tree. If you select a BST, then a balanced tree will be assigned to you for Program #4 (in Java). *** If you are implementing a balanced tree, you will <u>not</u> be expected to implement an individual removal operation (we will not be removing individual characteristics in a balanced tree!).

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.