Programming Assignment #3
CS 202 Programming Systems

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

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!

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

After an exhaustive day at the hospital with my daughter and finally off pain medicines, she is recovering by playing Halo. It could be Halo or other games, where good versus evil fight to save the world. With Halo we have Spartans and Covenants battling each other for victory to save humanity. Weapons include the magnum, DMR, rocket launcher and many more. There are multiple modes. There is campaign, fire fight, and forge modes. Campaign means that there is a story where Spartans fight against the Covenants. In fire fight mode, swarms of Covenants fight against Spartans in a smaller environment. In forge modes is where players create their own map and structure. Obviously data structures are

extensively used in these types of games to manage the weapons and the various covenant species who are an alien race.

Program #3 – Algorithm

Your job is to manage the data behind the scenes for popular computer games. You can pick Halo or another game where there are multiple types of devices (or weapons) and multiple types of players. The requirement for quick access of the data when requested. As a player plays a game, moves to a location, fires a weapon, and wins (or loses) a battle – the data structures must be used to keep track of how much ammunition each weapon has, and record the various strategies for saving the world. Although the game itself will not be played, we will provide some fundamental movement and operations (move to a location, fire, randomly receive the outcome of a firefight) that will be tracked. You may expand on these based on your familiarity with video games.

The primary goal is to create data types using operator overloading. You will need data types of weapon, and arsenal (the collection of weapons available); you will need a data type to track the player's movements and strategy (history of wins) along with their current location.

Program #3 –Data Structures

There will be three collections of data with this assignment using the following data structures; although this may vary from the actual Halo game itself – the following are the required data structures.

1. Balanced binary search tree of Spartans
2. Each node in the tree must contain a data structure for the Spartan's history. This should be implemented as a doubly linked list
3. Each node in the tree will also contain a particular Spartan's arsenal of weapons, stored as a hash table for quick access with their current ammunition available.

Please use of external data files for the weapon information pre-loaded into the data structures (by the constructors).

Provide full implementation of the data structures, although it is not required for deleting an individual item from a balanced binary search tree.

Program #3 – Operator Overloading

The primary purpose is to support operators with your list and table ADTs: =, +, +=, ==, !=, the relational operators, and the ability to input/output data. Think about how these operators might apply to your classes. Will you use the + and += to add movement or a win? Certainly << will allow you to display a Spartan's history or weapons arsenal. The = operator could allow weapons to be added and the – and -= could use ammunition for a particular weapon. But, what about the equality operator? Will the == allow us to find if a weapon is available for use? 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). Don't forget your copy constructor!

Side note. You CAN now write your own STRING data type, but it can't be the only place you use operator overloading!). For each of your operators make sure to fully test them.

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.