**Program 3 Design**

**Overview of Program**

The goal of this program is to organize and utilize different monsters for a simulated battle. It will allow the user to add different versions of four different types monsters and determine various characteristics of those monsters and their battle stats and main attack. There will be four types of monsters: a dragon, vampire, werewolf, and zombie. Additionally, there will also be a separate catalog of all know attacks and defenses. A main focus of this program will be the implementation of operator overloading for all the core aspects of the program.

**Base Class and Derived Classes**

The program will be implemented utilizing Object Oriented Programming. The base class of the program will be a *monster*. The base class itself will be able to hold the monster’s name and information. There will be four derived classes from this base class, the four classes will be the four types of will be four types of monsters: a dragon, vampire, werewolf, and zombie. The *dragon* class will maintain information on what kind dragon it is and will also maintain a dragon specific attack. The *vampire* class will maintain information on what kind vampire it is and will also maintain a vampire specific attack. The *zombie* class will maintain information on what kind zombie it is and will also maintain a zombie specific attack. The *werewolf* class will maintain information on what kind werewolf it is and will also maintain a werewolf specific attack. There will also be some further individual data members for the derived classes to differentiate them.

**Operator Overloading**

This program will support a full implementation of operator overloading for the monster base class as well as the derived class monster types. The main focus of the operator overloading will used on the monsters themselves as well as their attributes in order to operate the battle simulation. All of the following operators (=, +, +=, ==, !=) will be overloaded for these classes in order to alter and set the values for their attributes and other information. They will be implemented in both the monster and base class to ensure that no information is lost or mishandled within the program.

Additionally, both the relational operators will also be used to further simulate the monster battle by comparing their specific values as well as the input and output operators to facilitate the entering of the monster information as well as display the monster information to the user as needed from the data structures.

**Data Structures**

The program will be implemented using two data structures, a Binary Search Tree, and an Array of Doubly Linked List both created by their own node classes and managed by a singular list class. The binary search tree will consist of all of the different monster’s that are added to the program. They will be sorted by the name of the monster and will created utilizing a monster class pointer to utilize dynamic binding. Each of the nodes will contain one monster and house all of its information along with its monster identifier.

The array of doubly linked lists will be used to store the monster’s attack and defense information. Each index of the list will represent one of the monsters. Each of the monsters in that specific array will have a unique monster identifier linking it to the specific monster that is housed in the BST. This will allow the each of the nodes inside of the doubly linked lists to store the monster’s information and provide access to the client program if the information is desired.

The list class will manage both of the data structures. This class does not contain the information that is stored in each node, it only manages it. Both the root of the binary search tree and the array of head pointers will be housed in this node. This necessitates the use of setters and getters construct the binary search tree with its left and right pointers as well as in the creation of the doubly linked list. The functions of the list class will be Add, Remove, Display All, Display Type and Retrieve, there will be two versions of each of these functions, one for the binary search tree and the other for the array of doubly linked lists. Some of these functions will be supplemented with the operator overloading done with the monster classes to facilitate its use within the program.

**Summary**

This program will be created through the use of Object Oriented Programming and with the implementation of operator overloading in mind. There will be a base class named monster that has 4 derived classed, each being their own specific type of monster. The BST and Array of DLL’s classes will be managed by a list class with a “has a” relationship and will store different information regarding the monsters. Utilizing single inheritance, the program will create a systematic process for the user to access the monsters and their information and simulate the battle of the monsters using operator overloading.