Programming Assignment #2
CS 302 Programming Methodologies
and Software Implementation

***** Remember the Background Information still applies *****

*****This program is about Generic Programming *****

Program #3 - Goals

This term, the key idea is to break down the problems into small pieces and assign those responsibilities to individual classes. For each assignment, your job will be to build relationships between abstractions. 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 applying the concepts from Lab 4 and 5 with **templates** and **operator overloading** as the new syntax for your solution!

Templates are a great way to write our code once and use it for any data type desired. This does assume that the proper operators are overloaded so our underlying types used by the templates will perform properly. With this assignment, we will be implementing a List abstract data type where we will hide the data structures in a class and use templates to allow flexibility in the underlying data type. To that these data types will behave properly with the template software, you will need to support a full set of operators. **Exception handling** is also vital for handling for erroneous situations when working with operators – so the concepts we have been covering will all fit together!

Program #2 – Background

Everyone seems to love their animals. Once the pandemic hit, I never saw so many people walking their dogs! At one point, I had 5 dogs, 5 cats, a guinea pig, and a Greek tortoise. Definitely, it was a zoo! Now I only have three dogs and a big pile of fish.

You have decided to research how people interact with animals. What you have discovered is that there are many forms of interactions that we have. Of course, there are pets (such as cats, dogs, fish, etc.). But there are also other interactions, such as service or working animals (such as companion dogs) where we build relationships with those animals on a different level. Other animals are bred to

compete – either for top awards, agility, or dressage (a form of horse riding that is performed in exhibitions or competitions). The list goes on and on!

First Progress Submission: The core hierarchy

The core of your assignment is to create a way for the user to collect information about a variety of different types of animals and how they impact their owner or breeders' lives. You need to pick three different types of relationships we have with animals (e.g., pets, working, competitive, breeding, etc.) derived from a common base class. Push up the common elements of the three different types of animal relationships. Pick 2 or 3 items of interest that you would like to keep track of for each type of relationship along with information about those bonds that are created. For example, with a pet we would keep information about the species, the breed, and possibly age, temperament, shedding, etc. For an animal that is a working companion, we would want to know the species and breed, but also the type of job that they perform and how long they have been on the job. *Start with this hierarchy – any data structure you want for a collection of data should come from the STL!*

Second Progress Submission: Creating the Application

You will be creating a Doubly Linked List using templates for this assignment. This data structure should be used to keep track of a collection of animals that exist in a particular category (e.g., all of the information you have gathered about people's pets). So, if the user has pets, then they can learn about what types of bonds can be created with those pets and what pets to avoid. You should not be mixing different data types of animal relationship together in a given data structure, which means you first need to find out from the user which type they are interested (eg., pets, working, competitive, breeding, etc.).

You will be implementing a Doubly Linked List in this programming assignment. It must be implemented using templates – so both the Node class and the DLL class need to be class templates, much like you experienced in Lab4. This means we will implement the code only once to support each collection of animal relationships. As usual, all repetitive methods for the DLL must be implemented recursively.

The information in the Doubly Linked List must be "sorted" using overloaded operators for the core hierarchy data comparisons by the final submission!

For the second progress submission, progress should be shown on data structures implementation and towards being able to race. But a completed implementation is not expected until the finished due date.

Applying Operator Overloading

After Lecture and Lab during Week 5, it will be time to begin adding operators for you three types of animal relationships. Operator overloading is necessary so that your Class Template (the Doubly Linked List class) can be truly written independent of the underlying data. **Support the assignment, relational and equality operators, and I/O (<< and >>).** Support one of the binary arithmetic operators (such as + and +=) and the corresponding compound assignment operator that goes along with the operator you select. **Here is a summary of the operators to overload:**

1. Assignment
2. Relational (< <= > >=)
3. Equality (== !=)
4. I/O (<< >>)
5. One Binary arithmetic (e.g., +) and compound assignment (e.g., +=)

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 in each class). Don't forget your copy constructor and assignment operators for any class that manages dynamic memory! *The Powerpoint slides do have examples of how to overload each of these operators!*

For the second progress submission, complete at least 4 of the assigned operators.

Syntax REQUIRED for this assignment:

- **Exception Handling**
  - When something unexpected is experienced within an operator, we can't return success or failure. An exception must be thrown and handled by the client program.
- **Use the string class**.
  - Two char *'s should be used somewhere in a hierarchy (base-and-derived relationship). All others sould be strings.
- **Use at least one data structures from the STL (e.g., Array, List, Forward List)**
- **The DLL and Node class should be implemented using class templates**

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)?
- Avoid a void return type, with the exception of the assignment operator

b) Should it 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?
- Remember to never pass a class type by value

d) Remember to pass as a constant reference whenever possible.

e) Is the first operand always an object of class?
- If so, then it should be a member function.
- If not, then it should be a friend function.

f) 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.