Program #2
CS 202 Programming Systems

***** Make sure to read the Background Information first!**
It applies to all programming assignments this term***

****THIS IS NO DESIGN WRITEUP and NO UML DIAGRAM for Program #2 ****

Program #2 – Purpose

In our first programming assignment, we experimented with the notion of OOP and breaking the problem down into multiple classes building a relatively large system for a maze-game. The purpose of that assignment was to get to know how we can create classes that have different relationships. In program #2 our focus shifts to getting experience developing a hierarchy that can be used with dynamic binding – so that the application program can use one line of code to call one of many functions. We should still strive to create an OO solution, but now with the addition of dynamic binding. To get the full benefit of dynamic binding, we will look at a smaller problem with a predefined design. You are allowed to alter the design indicated below.

Program #2 – General Information

Have you ever noticed when you talk to people, you may use different forms of English (or you may use even a different language!) depending on the situation. For example, when I text, I only mention the most important words ("at home" or "leaving work") rather than saying "I am at home." Or "I am now leaving work"). When I talk to my daughter, I use words like "cool" but when I talk to my colleagues, my language is much more formal.

Program #2 – Building a Hierarchy using Dynamic Binding

For your second program, you will be creating a C++ OOP solution to support at least three different ways to talk with friends, family, and coworkers. The following represents three possible choices.

- **Texting:** The English is abbreviated ("thx" instead of "thanks", "cu" instead of "see you"). Each of us has our own abbreviations that we like to use. Support at least 10 different abbreviations using an external data file.
- **Formal:** In this case, we want to make sure there is one space between each word and two spaces after a period, exclamation point or question mark. Sentences should begin with a capital letter.
- **You pick the third choice**. It could be (a) providing a translation to different language (for 10 or more words), (b) informal such as when you talk with a family member (mapping some words to shorter abbreviations – like "M" for "mom"), or (c) when you talk to a toddler – making the language simple and easy to understand (i.e., having substitutions available for large words). Use an external data file to

assist with any translation needed from what the user supplies. This may be limited to just 10 combinations of words

The purpose of this assignment is to use and experience dynamic binding. The requirement for this application is to have at least **three DIFFERENT types of classes**, derived from **a common abstract base class**! To use dynamic binding, there needs to be a self-similar interface among the derived class methods. In this case, for all types of communication the user would need to input the text, display the resulting text translated into this format, modify the translation rules (e.g., for texting decide that "BRH" means to "Be Right Home"). Add one additional function besides these three. In the real world, there will be some differences as well, although there shouldn't be too many. **Add at least one method that is different so that you can experience how to resolve such differences.**

Program #2 – Data Structure Requirements
With program #2, our data structure should be of POINTERS to the ABSTRACT base class and then with upcasting cause each to point to the appropriate type of communication. Implementation of the required data structure(s) requires full support of insert, removal, display, retrieval, and remove-all. Remember all character data must be stored using dynamically allocated arrays of characters. Consider creating an abstraction for this!

You will need to implement data structures for the following. **ALL repetitive algorithms should be implemented recursively to prepare for our midterm proficiency demos!**

1. **Original text entered by the user** – a linear linked list of arrays of characters where each word is stored in its own node.
2. **Then, the application needs to store objects of a variety of different communication types** – Use a doubly linked list of abstract base class pointers pointing to the different objects that represent the user's forms of communication. For example, the user may want to text 3 people different messages, send an email to a family member, and write a memo to their boss. Have both a head and a tail pointer.

Program #2 – Important C++ Syntax
Remember to support the following constructs as necessary:
1. Every class should have a default constructor
2. Every class that manages dynamic memory must have a destructor
3. Every class that manages dynamic memory must have a copy constructor
4. If a derived class has a copy constructor, then it MUST have an initialization list to cause the base class copy constructor to be invoked
5. Make sure to specify the abstract base class' destructor as virtual

IMPORTANT: ***OOP and dynamic binding are THE PRIMARY GOALS of this assignment!***