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. 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. To get the full benefit of dynamic binding, we will look at a smaller problem with a predefined design.

Program #2 – General Information

We are in an unusual time in history with our “Stay Home, Stay Safe” mandate. For me, I have been so swamped with work, that the last five weeks “at home” have flown by and I don’t see an end in sight. There is simply a mountain of work to do! But, there are moments when I look outside at the beautiful trees and the blooming dogwood tree, that I start imaging all of the activities I will do once we start opening up Oregon again. My top picks are to go hiking on the trails, but I also want to go out to dinner in downtown Portland, and even visit the animals at the Zoo. I’ve started a list of all that I want to do! How about you?

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 types of activities or projects that you want to do once the Stay Home, Stay Safe mandate has been lifted. Create a base class for general information that is common among the three classes. This will be an abstract base class since we would never work with a general activity or project.

Then, create three different derived classes for the specific types of activities or projects. Each type should have similarities with each other – so don’t pick ideas that are vastly different. There needs to be a self-similar interface. So for example, with my three examples (hiking, dining, and zoo) my functions will be to input the information about the activity, display it, set a date for when I want to go, determine how long it will take (hiking is a half day activity while dining is 1 hour), etc. For all of them, you need a title or name, a priority level, and the length of time the activity or project takes, and if this activity is an independent one or one that is done with a friend. The priority level will let the user indicate if they want to do this immediately or if this is for later. For each of your three activities, pick one function that is different so that you can experience RTTI in practice; this is a requirement.

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 but the implementation of the methods would need to be different in some way for you to make the most of this experience. **Make sure to find at least one method that is different (in name and/or argument lists) so that you can experience how to resolve such differences with RTTI.**

Program #2 – Data Structure Requirements
The real beauty of dynamic binding is that we can have a data structure be a collection of different data types! There needs to be ONE data structure of base class pointers. This data structure will keep all activities or projects of all types, using upcasting. The data structure will be a **Doubly Linked Lists of arrays**, where each array element contains a base class pointer, organized (sorted) by title (or name). The Doubly linked list should be organized by the most important items (so the first node contains all of the activities I want to do first – my top priority)! Implementation of the required data structure(s) requires full support of insert, removal, display, retrieval, and remove-all. There should be a way to be able to display just those items that are top priority!).

ALL repetitive data structure algorithms should be implemented recursively to prepare for proficiency demos!

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 (but don't make it a pure virtual function)
6. It is expected that you will experience RTTI with dynamic_cast in this assignment

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