Programming Assignment #1
CS 202 Programming Systems

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

We do not accept late work beyond the late due date.

Always backup your files PRIOR to using tar
Double check that the arguments specified with for tar are correct!
Late penalties will apply to all submissions incorrectly archived/uploaded

Background:
When beginning with this project, the first thing to keep in mind is that we are no longer working on CS163 programs! In CS163 we were concerned about creating Abstract Data Types and the class construct facilitated this. Instead, this term we will be focusing on how to create Object Oriented Solutions. An ADT may be part of that solution – but it certainly shouldn't be the primary focus. Instead you want to strive for classes to have specific "jobs" and have classes derived from more general classes, whenever appropriate. We will be working in situations where there are multiple classes, so you will want to focus on dividing the design into smaller components that have specific jobs working together to solve the problem.

Every assignment this term needs to have at least 5 classes. With these, think about how to design the classes such that they reduce the amount of work another class needs to do. The idea is if we have "robot" like classes doing the smaller tasks or "jobs", that by the time we get to a larger class that has more to manage – it will have little left to do! We can achieve this by delegating. Often the over-use of "getters" can cause the opposite to happen – and instead of delegating the managing class has to fundamentally do all of the work itself.

Overview:
Drones have become very popular. They are now used for wide variety of activities from military to entertainment. Do you have a drone? Maybe one or two of them? Well, let's now go drone racing through an obstacle course. Maybe we should start a club?!

Drones have many different kinds of actions. You can take off and land. Drones can move in three dimensions: yaw, pitch (e.g., up and down), roll (e.g., rotation front to back). Drones should never go above the approved ceiling for flying in a particular area. They also cannot fly within 5 miles of an airpor or near emergency response efforts.

Program #1

For Program #1, you will be creating an object oriented program that will simulate the notion of drone racing through an obstacle course.

The first step will be to think about breaking this down into a series of classes and create them independent of the entire problem. You will want to not only think about what a drone is but also what an obstacle course is. Some relationships should be hierarchical, others can be containment. With hierarchies always push the common elements up to the base class. And Nodes will need to be classes instead of structs.

Let's begin by talking about the obstacle course. There needs to be three different types of obstacles. Here are some ideas:

1. Goal posts where the drone flies over (or under) a bar.
2. A black hole where if a drone gets to close, gravity will send back to the beginning.
3. Touch and Go, where the drone must land on a particular spot before continuing.
4. Circular, where the drone flies all the way around an obstacle before continuing.

You must implement two of the above list of obstacles and add one other of your own design. Your version must be uniquely your concept.

Then once you have decided upon the desired classes and relationships, then start mapping out operations that will control the drones and build the obstacle course. Avoid classes with only setters and getters! It is during this step you need to consider how to keep drones from colliding! Support should be in place such that a drone will not let itself enter a restricted area.

The following represents some ideas on the design – the first step is to plan what classes might make the most sense! Some of the basics that are part of an OO program **could** be:

1. Controller class – which controls the movement of a drone
2. Location class – where the drone or obstacle exists
3. Three specific types of obstacles derived from a general obstacle or location class
4. Drone class

Anything that is similar between these or other classes that you write should be pushed up to be part of a base class. For example, classes that manage collections of items may be derived from a common base class that manages the collection. Keep classes small and functions small. A large class or function means that the problem has not yet been broken down into its basic components (objects).

Data structures
The drones in the race should be handled with a doubly linked list. You will need to consider where the front running drones are located (head or tail) and what happens when there are severe collisions (that take drones out of the race). You will need both a head and a tail pointer.

The obstacle course should be implemented with a Graph. Your graph should be implemented using an adjacency list which is a **dynamically allocated array of head pointers to edge lists**. Each head pointer points to a LLL of straightaway's and adjacent obstacles. There should be more than one path from the start to the finish line through the set of straightaway's and obstacles!

All traversal functions should be implemented **recursively**.
All arrays must be implemented dynamically with a non-constant size.

Have fun with this!