Programming Assignments #1 and 2
CS 202 Programming Systems

Submit your program to the D2L Dropbox

We do not accept late work beyond the late due date. There are no exceptions.

ALWAYS make a backup of your programs before using tar
Double check that the arguments specified with for tar are correct!
***A 5% penalty** 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:
We have now boarded a starship ready to take us to a neighboring planet in a galaxy far far away. Just like with our galaxy, planets rotate around a sun, have gravitational pull, are affected by the distance from the sun, and may or may not have atmosphere. Our path of travel will be affected by the gravitational pull of neighboring large planets, suns and even moons. Our goal is to search for a planet that will sustain life (e.g., have an atmosphere) and where the distance from the sun is just right (e.g., too close and we will burn up – too far and we will freeze). Living on just the dark side of a planet will not be sufficient. The first to find such a planet will save the human race!!

Details for Program #1 and 2:

Your program will need the following components:

1. A galaxy!
    a. You may start small with a galaxy consisting of at least two solar systems with at least 5 planets each.
    b. For each planet, there may be different types (terrestrial (inner) planets and gas giants (outer planets)). Terrestrial planets have solid surfaces.
    c. For each planet, the distance from the sun, the size of the planets, and the information about the atmosphere must be affected by a random number generator. It should not be predictable and it should not be the same each time you run your program!
    d. The remaining information can be read from an external data file (e.g. such as the planet's name and the sun that the planet is orbiting).
    e. Information about the solar system should not be hard coded!

2. A small spaceship
    a. To get the spaceship where we want it to go, we need to know how fast it needs to go and take the gravity of large bodies (like planets and suns) into account as it will bend the flight of our spaceship. Use your imagination and have some fun with this!

Program #1 – Setting up the Galaxy

Although you need to design the entire program (there is only one design for the program 1-2), you will want to break down the implementation. With program #1, you will get started with:

1. Creating the galaxy that contains solar systems
2. Creating Solar Systems that contain one sun and a set of planets
3. Allow a planet to have zero or more moons orbiting
4. Allow planets to be either habitable with the appropriate atmosphere and distance from the sun, or not
5. Some planets are outer planets, and others will be inner planets.
    a. An outer planet is a planet that isn't solid; an outer planet will likely have gas, moons and rings
    b. An inner planet is a planet that is solid such as a terrestrial planet; it has no or few moons.
6. You may add additional characteristics!

The primary job for program #1 will be to create a galaxy of planets and then determine if each planet is habitable or not.

1. At least 5 classes are required.

2. Have both containing and hierarchical relationships supported.
3. Create a data structure of a dynamically allocated array of solar systems, with a doubly linked list of all of the planets in that solar system arranged by distance.
4. At least one of your functions working with the data structure must be implemented using recursion

Program #2 – Traveling through Space

In the next level, create the spaceship and have it begin traveling through your galaxy investigating the various planets in search of one that can support life. Realize as part of your spaceship that it won't always travel in a straight-line due to the influence of neighboring planets. We also don't want to run into a planet, so knowing where the planets are in the solar system is going to be important. Now your planets will need to orbit around their sun.

Now we must include:

1. The data structure for the galaxy must be an array of circular linked lists (CLL) where each CLL keeps track of the location of the planet in its orbit around its sun much like a graph's adjacency list. Each node needs to contain a pointer to a base class so that it can point to any kind of planet using dynamic binding and upcasting.
2. **At least one function must be pure virtual so that you experience abstract base classes in your design.**

To make this Object Oriented:

You will want to first think about breaking this down into a series of classes and create them independent of the data structure. Pay close attention to the Course Outline for the due date for the design and programs.

Here are some suggestions to start with:

1. A Galaxy has Solar_Systems
2. A Solar_System has planets
3. Planet is its Habitability
4. Inner_Planet is a Planet
5. Outer_Planet is a Planet

Anything that is similar between these or other classes that you write should be pulled 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.

AS SOON AS YOU ARE DONE WITH program #1, begin on the rest to get program #2 completed!!!!!