CMPS 2143 Object Oriented Programming
Program 4: 100 points
DUE: Wednesday, Nov. 15, 2017

Purpose: To apply polymorphism and overload an operator (=); and to use static variables.

Problem: *Space Fleet 2.0*

NASA2 was so thrilled with the success of SpaceCraft 2.0, they want to send *many* probes and ships into space. As in program 3, you will still need to read in asteroids from a file. You will *also* read a file that has information on spacecraft in the fleet and put it in an array of pointers to spacecraft. See sample input below.

The space probes go out, visit a certain number of asteroids, and determine if they are composed of precious metals. If so, the probe *copies* the asteroid's data to a *now shared static* internal list of precious metals asteroids declared in the *SpaceCraft* class. So now, all probes and spaceships *share* this list! Spaceships will search through the joined precious metals list and collect a certain number of precious metals asteroids less than a certain size.

Note: The spacecraft work together and share four static variables: the *total number of asteroids to visit* and the *total number of asteroids visited*, the *total number of asteroids to collect* and the *total number of asteroids collected*. Keep in mind the probes and ships still keep track of how many asteroids they themselves visit or collect and this is what they will "signal" Earth, that is print.

Required Concepts to Apply:

- Move the list of precious metals asteroids to the SpaceCraft class. Make it `static`, so it will be shared amongst all spacecraft.
- In the SpaceCraft class, declare four `static` variables to keep track of values: the total number of asteroids to visit and the total to collect and the total number visited and total collected.
- Make sure you have a copy constructor in Asteroid class.
- Write a `toString` method in *all* your classes that formats and returns a string of the values of the member data formatted the way you want the main to print them. NOTE: Don't have SpaceCraft include the precious metals asteroid list.
- Add a string member data called `name` to SpaceObject to indicate the type of the object and make sure the `toString` method returns this type for all all subclasses. Modify all the constructors to set this field. Add a get method to SpaceObject to return this data. The member data and method will be inherited.
- In main, use a typedef and declare an array of pointers to SpaceCraft.
- Overload the = operator for the Asteroid class, and use it in any method that needs to assign values from one asteroid to another, e.g., `alist[i] = asteroid;` will NOW be a legal statement and will replace statements like `alist[i].setWeight (asteroid.getWeight());`
- You will need to add a `toString` method to the `AsteroidList` class, as well.

- Add a `getList( )` method to the SpaceCraft class, that calls the `toString` method of the AsteroidList, to return *one huge string* of precious asteroids to print out later. This is essentially a dump method.

Input: See Program 3's lengthened file and a new file called "Spacecraft.txt." A sample file looks as follows:

```
7                   //number of spacecraft
35                  //number of asteroids for the probes to visit
10                  //number of precious metals asteroids to collect
6                   //largest asteroid to collect
P   0   0           //probe at (0,0)
P   50 50
P  100  100
P  100  0
P  0     100
S   0     0         //ship at (0,0)
S   80   80
```

Output: Your program should print out

- "status reports," that is, each probe's state and each ship's state at the beginning of the missions, as well as the type of craft it is. (*Use a for loop and polymorphically call the toString method.*)
- A non-member function in main should then print the probes' combined visited precious metals asteroid list in a tabular format. (*That means the toString method in AsteroidList formats the data!!!*)
- Then, the program will print out the "status reports," that is, each probe's state and each ship's state at the end of their missions.
- Finally, a summary (the values of the 4 static variables) is printed.

Turn in:

1. My input data files for program 4.
2. An additional input file for spacecraft data of your own. Try to collect more asteroids than is possible.

3. Console output from two execution runs.
4. Two output files.

5. Class files (.h and .cpp)
6. Main program file.