CS162
Programming Assignment 2 - Ship Class

The purpose of this assignment is to give you an introduction into creating your own classes and using objects of that class.

Overview

This program is part one of many that will implement an Asteroids-like game. Future assignments will add additional functionality to this class, and the class will be used in a larger project. Taking the time to thoroughly test and debug this code now will save much time later.

Details

Write all the code necessary to implement the Ship class as given below. Note that it makes use of a Vector type which you'll need to define. A Vector should be a struct that contains an x and y coordinate (both floats).

Note: Vector is a name already defined in std so you might want to pick a different name. If you use Vector, do NOT include the line "use namespace std" at the top of your code. Instead, preface any cout with std::. Only the shiptest.cpp file will need to access cout.

Your class definition and implementation should be in separate files.
When complete, you should have the following four files:

- vector.h - definition of the Vector struct (which contains an x and y coordinates, as floats)
- ship.h - class definition (interface)
- ship.cpp - class implementation
- shiptest.cpp (includes a main that tests the ship class code)

Don't forget to include the proper preprocessor directives in your header files!

Following is the class definition (belongs in ship.h). Function details are below.

```
class Ship {
private:
      Vector maxLocations;     //maximum allowable values for location
      Vector location;         //current location (x,y)
      Vector velocity;         //current velocity (in pixels/frame)
      float angleDeg;          //angle ship is facing, in degrees
      float radius;      //gross radius of ship (for collision detection)

public:
      Ship();
```

```
        //==============================================
        //mutators
        void setLocation(float x, float y);
        void setVelocity(float velocityX, float velocityY);
        void setAngle(float angDeg);

        //=============================================
        //accessors
        float getRadius();
        Vector getLocation();
        Vector getVelocity();
        float getAngle();

        //=============================================
        //others
        void rotateLeft();
        void rotateRight();
        void applyThrust();
        void updateLocation();
  };
```

The functions should do the following:

Ship()	initialize all member variables to 0, except maxLocations, which should be 1000,1000, and radius, which should be 5.
All Mutators	set associated member variables to the values given. Validate location values and truncate them if necessary to remain in bounds (from 0 to maxLocation values)
All Accessors	return copies of requested data

rotateLeft()
rotateRight()
	Subtract 1 from angleDeg (when it goes to -1, make it 359)
	Add 1 to angleDeg (if it exceeds 360, set it to 1)

applyThrust()	Simulates firing the engine. Changes the current velocity based on the angle the ship is facing (it may be facing a different direction than it's travelling). Math is as follows:
	`float forcex= cos((angle-90)*PI/180) * .005;`
	`float forcey= sin((angle-90)*PI/180) * .005;`
	`velocity.x = velocity.x + forcex;`
	`velocity.y = velocity.y + forcey;`
	Note; the -90 is there because an angle of 0 is usually pointing to the right and we want it to be pointing up.
	The PI/180 converts our angle, which is in degrees, to radians, which is what the sin and cos functions want. You'll probably need to define PI.
	The .005 is the thrust the of the engines. Obtained via experimentation. May be tweaked in the future. Should probably be a constant.

updateLocation()	Updates the location based on the current velocity. (adds the velocity vector to the location vector)

	If any resulting value is below 0 add the associated maxLocation value to it If any resulting value is greater than the associated maxLocation value, subtract the associated maxLocation value from it. These "corrections" make the ship reappear back on the opposite side of the window if it goes off the edge.

Additional functions to implement more features (drawing it on the screen, collision detection, etc..) will be added in future assignments.

You should write and test each function thoroughly before moving to the next function. main() will consist of code that simply tests your class code, calling functions with test values and displaying the results. Thorough testing now will save you time later!!!!

Very Important Stuff

All programs should follow the class's coding conventions.
Submit the following:

- A zip file containing the four source files you created and your executable