Physics 307 Homework 6
Due Thursday, 25 October, at 5 PM

In this homework assignment, you will use your orbit program to simulate a highly elliptical orbits, and then a binary star.

1. Consider a highly elliptical orbit such as a comet. At aphelion, Halley's comet is 35.1 AU from the Sun and has a velocity of 0.193 AU/year. Use your program to simulate the orbit of Halley's comet. How close does it come to the Sun at perihelion, and how fast is it traveling there?

   About how many timesteps are required to simulate this orbit reasonably accurately? Compare this to the value for a circular orbit that you determined

2. **Extra credit:** Can you think of any way to mitigate this and simulate the orbit with fewer timesteps? Implement it.

   last week. What makes cometary orbits so much harder to simulate accurately?

3. If you're not familiar with Kepler's laws of orbital motion, look them up. Does the behavior of your simulation reflect Kepler's second law (qualitatively; you do not need to measure areas!)

4. Now, modify your program to simulate a binary star system with two stars of unequal mass– that is, two objects responding to each other's gravity.

   Hints:

   - If you measure mass in solar masses, G still has a value of $4\pi^2$.
   - You will now need to use $\vec{F} = m\vec{a}$ and $\vec{F} = \frac{Gm_1m_2}{r_{12}^2}$ together to find the accelerations.
   - Your count of dynamical variables has now doubled to eight:
     - $x_1, y_1, v_{x_1}, v_{y_1}$ for the first star
     - $x_2, y_2, v_{x_2}, v_{y_2}$ for the second star
   - The leapfrog prescription is still the same:
     (a) Evolve the position variables (all four of them!) forward by $dt/2$
     (b) Evolve the velocity variables (all four of them!) forward by dt (this is the hard part)
     (c) Evolve the position variables forward by $dt/2$ again

     Your cut-and-paste skills will come in handy here, as you will be typing multiple copies of very similar code. But be careful of editing errors when doing this!

- The radius vector that appears in the differential equations is now the *separation vector* between the two stars. The origin no longer plays any special role in the dynamics.
- Since the x and y that appear explicitly in the differential equations come from the components of $\hat{r}_{12}$, they will become $x_1 - x_2$, etc.

5. Important: if you don't want your simulation to "drift" out of the viewport, then you will want to ensure that the total momentum is zero. See the end of this document for how to do this.

   Run your simulation, and monitor conservation of total energy to ensure that it is behaving properly. You now can't compute energy per unit mass, since you have two masses; the total energy will be

$$E = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 - \frac{Gm_1 m_2}{r_{12}} \tag{1}$$

6. Now, finally, modify your code to simulate the gravitational interactions of *three* bodies. This is no more complicated than two; you just have some copy-paste work to do, since each object now feels the force from *two* neighbors, rather than just one.

   Play around with what you can create – make things move in three dimensions, etc. Note that if two bodies get too close together, they will experience a very large force that is probably too big for your timestep to accurately simulate. Can you make a stable sun-planet-moon system?

Avoiding solar system drift

The total momentum of a system of two stars is

$$\vec{p} = m_1\vec{v}_1 + m_2\vec{v}_2$$

This gives a center-of-mass velocity of

$$\vec{v}_{\text{com}} = \frac{m_1\vec{v}_1 + m_2\vec{v}_2}{m_1 + m_2}$$

By calculating this value and subtracting it from each of your objects' initial velocities, you ensure that the total center-of-mass velocity (and thus the total momentum) is zero, and your simulation won't drift. This can be done with code like the following:

```
double vxc,vyc; //center-of-mass velocities
vxc = (m1*vx1 + m2*vx2) / (m1+m2);
vyc = (m1*vy1 + m2*vy2) / (m1+m2);
vx1 = vx1 - vxc;
vy1 = vy1 - vyc;
vx2 = vx2 - vxc;
vy2 = vy2 - vyc;
```