Question 1

The data files used for this assignment start by stating the number of CelestialBody objects in the data file. Suppose this were not the case; could you still run the simulation if all other data were present? Briefly explain how you could still create an array of CelestialBody objects of the same size when reading the file.

You could still run the simulation in a couple of ways. You could create an arraylist of celestialbody objects, and then convert this arraylist into an array. You could also count the number of data entries and use this to determine how many objects there are in the data. Then you would build an array of that size like before.

Question 2

If there are n CelestialBody objects, how many many total times will the code have to execute the calcForceExertedByX method per time step (that is, per iteration of the outer loop of the main method) of the simulation? Your answer should be in terms of n. Briefly explain your answer.

CalcForceExertedByX will be executed n(n-1) times. CalcNetForceExertedByX calls the method n-1 times and the main method of the simulation calls CalcNetForceExertedByX n times.

Question 3

In terms of totalTime and dt, how many total time steps (that is, iterations of the outer loop of the main method) will there be in the simulation? Briefly explain your ansswer. Based on this, would increasing the value of dt increase, decrease, or have no impact on computational resources necessary to run a complete simulation?

TotalTime / dt = 1577.88. This means there will be 1578 time steps, because the loop will start at zero and continue to iterate until the variable in the for loop, t, is greater than dt*1577.88. t will end at dt*1578;

Question 4

dtwas initially set to 25000.0. Change this value to 1000000.0 (one million) and run the simulation again. You should see behavior inconsistent with what is expected for the simulation. Briefly explain why increasing the value of dt could cause this behavior.

When dt is changed to 1,000,000 the simulation runs much faster. The planets move faster and the simulation runs for a shorter period of time. Having a larger dt increases the velocities assigned in the update function, because they are related to dt. This causes the planets to move faster. Having a larger dt also causes the for loop in the main method to end faster, because it takes fewer iterations to reach the max t value.