```
    1: #include "CelestialBody.h"
    2: #include <memory>
    3: #include <vector>
    4:
    5: int main(int argc, char *argv[]) {
    6:
    7:   if (argc != 3) {
    8:     return -1;
    9:   }
   10:   std::string simulation_Time(argv[1]);
   11:   std::string step_Time(argv[2]);
   12:   std::string::size_type sz;
   13:
   14:   cout << "Simulation time: " << simulation_Time << endl;
   15:   cout << "Step time: " << step_Time << endl;
   16:
   17:   double totalTime = std::stod(simulation_Time, &sz); //, &sz
   18:   double stepTime = std::stod(step_Time, &sz);        //, &sz
   19:   double simTime = 0.0;
   20:
   21:   std::string planets, rad;
   22:
   23:   std::cin >> planets;
   24:   std::cin >> rad;
   25:
   26:   int numPlanets = std::stoi(planets.c_str());
   27:   double radius = std::stod(rad.c_str());
   28:
   29:   cout << "Number of planets: " << numPlanets << endl;
   30:   cout << "Radius: " << radius << endl;
   31:
   32:   std::vector<std::unique_ptr<CelestialBody>> bodies;
   33:
   34:   for (int i = 0; i < numPlanets; i++) {
   35:     CelestialBody *temp = new CelestialBody();
   36:
   37:     cin >> *temp;
   38:     temp->setRadius(radius);
   39:     temp->setPosition();
   40:     //cout << *temp << endl;
   41:
   42:     bodies.push_back(unique_ptr<CelestialBody>(temp));
   43:   }
   44:
   45:   sf::RenderWindow window(sf::VideoMode(winWidth, winHeight), "Solar System"
);
   46:
   47:   window.setFramerateLimit(60);
   48:   sf::Image background;
   49:
   50:   if (!background.loadFromFile("background.jpg")) {
   51:     return -1;
   52:   }
   53:
   54:   sf::Music music;
   55:   if(!music.openFromFile("2001.wav"))//make sure music file opens
   56:   {
   57:           return -1;
   58:   }
   59:   music.play();
   60:
```

```
   61:   //for displaying elapsed time
   62:   sf::Font timeFont;
   63:   if(!timeFont.loadFromFile("arial.ttf"))
   64:   {
   65:           return -1;
   66:   }
   67:   sf::Text timeText;
   68:   timeText.setFont(timeFont);
   69:   timeText.setCharacterSize(15);
   70:   timeText.setFillColor(sf::Color::White);
   71:
   72:   sf::Texture backTex;
   73:   backTex.loadFromImage(background);
   74:   sf::Sprite backSprite;
   75:   backSprite.setTexture(backTex);
   76:
   77:   while (window.isOpen()) {
   78:     sf::Event event;
   79:     while (window.pollEvent(event)) {
   80:       if (event.type == sf::Event::Closed) {
   81:         window.close();
   82:       }
   83:     }
   84:     window.clear();
   85:     window.draw(backSprite);
   86:
   87:     for (auto itr = bodies.begin(); itr != bodies.end(); itr++) {
   88:       window.draw(*itr->get());
   89:       timeText.setString("Elapsed time: " + to_string(simTime));
   90:       window.draw(timeText);
   91:
   92:       //cout << *itr->get() << endl;
   93:
   94:       double forceX = 0;
   95:       double forceY = 0;
   96:       //sum up the x and y forces on each body
   97:       for (auto itr2 = bodies.begin(); itr2 != bodies.end(); itr2++) {
   98:         if (itr != itr2) {
   99:           forceX += getForceX(*itr->get(), *itr2->get());
  100:           forceY += getForceY(*itr->get(), *itr2->get());
  101:         }
  102:       }
  103:       (*itr)->setForces(forceX, forceY);
  104:
  105:       (*itr->get()).step(stepTime);
  106:       // cout << *itr->get() << endl;
  107:       (*itr->get()).setPosition();
  108:       //cout << *itr->get() << endl;
  109:     }
  110:     window.display();
  111:
  112:     simTime += stepTime;
  113:     if (simTime >= totalTime) // end of simulation
  114:     {
  115:       break;
  116:     }
  117:   }
  118:
  119:   //debugging
  120:   cout << "\n\nFinal postitions:\n" << endl;
  121:   cout << numPlanets << endl;
```

```
  122:   cout << radius << endl;
  123:   for(auto i = bodies.begin(); i != bodies.end(); i++)
  124:   {
  125:           cout << *i->get() << endl;
  126:
  127:   }
  128:   return 0;
  129: }
```