```
 1: // Angel Zheondre Calcano
 2:
 3: #include <iostream>
 4: #include <string>
 5: #include <vector>
 6: #include <SFML/Graphics.hpp>
 7: #include <SFML/System.hpp>
 8: #include <SFML/Window.hpp>
 9: #include "space.hpp"
10:
11: using namespace std;
12: using namespace sf;
13:
14: int main( int argc, char *argv[] ) {
15:
16:   int n, i, ws ; double spcrdi; vector< body* > space ;
17:   cin >> n ; cin >> spcrdi ; ws = 900 ;
18:
19:   for( i = 0 ; i < n ; i++ ) {
20:     body *planet= new body( spcrdi, ws ) ;
21:     cin >> planet;
22:     space.push_back( planet )  ;
23:    }
24:
25:   RenderWindow window( VideoMode( ws, ws), "ps3 AC" ) ;
26:
27:   while (window.isOpen()){
28:
29:     sf::Event event;
30:     while (window.pollEvent(event)) {
31:       if (event.type == sf::Event::Closed)
32:         window.close();
33:     }
34:     window.clear();
35:     for( i = 0 ; i < n ; i++ ) window.draw(*space[i]) ;
36:     window.display();
37:   }
38:   return 0 ;
39: }
```

```
 1: // Angel Zheondre Calcano Ps3a N-Body Simulation
 2: #include <iostream>
 3: #include <math.h>
 4: #include "space.hpp"
 5:
 6: using namespace sf ;
 7:
 8: body::body(  double ss, double ws ) {
 9:   ceny = cenx = ss/(1e+9) + 200;
10:   winsiz = ws;
11: }
12: void body::center( double x) {
13:   ceny = cenx = x/(1e+9) + 200;
14: }
15: void body::setpos(){
16:   radfromsun = xpos/(1e+9);
17:   xpos = radfromsun + cenx;
18:   ypos = ypos/(1e+9) + ceny;
19:   //grav(radfromsun) ;
20: }
21: double body::getposx(){ return xpos; }
22: double body::getposy(){ return ypos; }
23: double body::getxvel(){ return xvel; }
24: double body::getyvel(){ return yvel; }
25: double body::getmass(){ return mass; }
26: double body::getrad(){ return radfromsun; }
27: string body::getfname(){ return fname; }
28: void body::setImage(){
29:   texture.loadFromFile(fname) ;
30:   sprite.setTexture(texture) ;
31:   sprite.setPosition(xpos, ypos ) ;
32: }
33: void body::newpos(){
34:   if( xpos > cenx && ypos <= ceny ) {
35:     xpos = xpos - xvel;
36:     ypos = ypos - yvel ;
37:   }
38:   if( xpos < cenx && ypos <= ceny ) {
39:     xpos = xpos - xvel;
40:     ypos = ypos + yvel ;
41:   }
42:   if( xpos >= cenx && ypos < ceny ) {
43:     xpos = xpos + xvel;
44:     ypos = ypos + yvel ;
45:   }
46:   if( xpos >= cenx && ypos > ceny ) {
47:     xpos = xpos + xvel;
48:     ypos = ypos - yvel ;
49:   }
50: }
51: /*int body::grav(double r){ //implement later
52:  //need grav pull to make circular motion
53:   if( r > 1 ) return -1 ;
54:   xvel = (Grav*mass*m2)/(r * r) ;
55:   return 0 ;
56: }
57: */
58:
```

```
 1: #include <cmath>
 2: #include <iostream>
 3:
 4: using namespace std ;
 5:
 6: int main( int argc, char *argv[] ) {
 7:
 8:   cout << sin(1) << endl  ;
 9:   cout << 500*sin(1*(M_PI/180) ) << endl  ; // use for rad to deg conver
10:
11:
12: }
```