```
using System;
using System.Collections.Generic;
using System.Linq;
using static Program.Constants;
namespace Structures {
    public static class Tests {
        public static bool MatrixTest() {
            // Scalar Arithmetic
            var i = new Matrix3(new Vector3(1,0,0),new Vector3(0,1,0),new Vector3(0,0,1));
            var a = new Matrix3(new Vector3(1,3,1),new Vector3(0,4,1),new Vector3(2,-1,0));
            if (i*i != i) {
                Console.WriteLine("i*i != i");
                return false;
            }
            if (i*a != a || a*i != a) {
                Console.WriteLine("a*i != a");
                return false;
            }
            if ((double)5 * a / (double)5 != a) {
                Console.WriteLine("5*a/5 != i");
                return false;
            }
            if (a + a != 2 * a) {
                Console.WriteLine("a + a != 2 * a");
                return false;
            }

            // Inverse (Also tests Determinant, Minor, Transpose_Cofactor)
            var a_inv = new Matrix3(new Vector3(-1,1,1),new Vector3(-2,2,1),new Vector3(8,-7,-4));
            if (Matrix3.Inverse(a) != a_inv) {
                Console.WriteLine($"Matrix3.Inverse(i) == \n{Matrix3.Inverse(a)} != \n{a_inv}");
                return false;
            }
            if (Matrix3.Inverse(Matrix3.Inverse(a)) != a) {
                Console.WriteLine("inv(inv(a)) != a");
                return false;
            }
            try {
                Matrix3.Inverse(new Matrix3(Vector3.zero,Vector3.zero,Vector3.zero));
                Console.WriteLine("No Exception on Inverse of Singular Matrix");
            } catch (DivideByZeroException) {}

            // Matrix-Matrix Multiplication (Also tests Transpose)
            var a_sq = new Matrix3(new Vector3(3,14,4), new Vector3(2,15,4), new Vector3(2,2,1));
            if (a * a != a_sq) {
                Console.WriteLine($"a * a != a_sq");
                return false;
            }
            return true;
        }
        public static bool VectorTest() {
            var a = new Vector3(2,3,6);
            var z = Vector3.zero;
            // Scalar Arithmetic
            if (a + z != a || z + a != a) {
                Console.WriteLine("a + z != a");
                return false;
            }
            if ((double)5 * a / (double)5 != a) {
```

```
60                  Console.WriteLine("5*a/5 != i");
61                  return false;
62              }
63              if (a + a != 2 * a) {
64                  Console.WriteLine("a + a != 2 * a");
65                  return false;
66              }
67              if (-a != z - a) {
68                  Console.WriteLine("-a != z - a");
69                  return false;
70              }
71              if (Vector3.Magnitude(a) != 7) {
72                  Console.WriteLine("Incorrect Magnitude");
73                  return false;
74              }
75              if (Vector3.dot(new Vector3(1,2,0),new Vector3(-2,1,0)) != 0 ||
    Vector3.dot(a,a) != 49) {
76                  Console.WriteLine("incorrect dot");
77                  return false;
78              }
79              if (Vector3.cross(new Vector3(3,-3,1), new Vector3(4,9,2)) != new
    Vector3(-15,-2,39)) {
80                  Console.WriteLine("incorrect cross");
81                  return false;
82              }
83              var a_u = new Vector3((double)2/7,(double)3/7,(double)6/7);
84              if (Vector3.Unit(a) != a_u) {
85                  Console.WriteLine("incorrect unit");
86                  return false;
87              }
88              var exp = new Vector3(1000,0,-100);
89              try {
90                  Console.WriteLine(Vector3.Unit(Vector3.zero));
91                  Console.WriteLine("Unit(zero) did not throw exception");
92                  return false;
93              } catch (DivideByZeroException) {
94              } catch (Exception) {
95                  Console.WriteLine("Incorrect exception");
96                  return false;
97              }
98              if (Vector3.PolarToCartesian(Vector3.CartesianToPolar(a)) != a) {
99                  var b = Vector3.PolarToCartesian(Vector3.CartesianToPolar(a));
100                 Console.WriteLine((a.x - b.x)/a.x);
101                 Console.WriteLine("Cartesian-Polar conversions failed");
102                 return false;
103             }
104             Vector3 c = null;
105             Vector3 d = null;
106             if (a == c || c != d) {
107                 Console.WriteLine("Null checks incorrect");
108                 return false;
109             }
110             return true;
111         }
112
113         public static bool BodyTest() {
114             var sun = new Body {
115                         stdGrav = 1.3271440019e20,
116                         radius = 6.95e8
117             };
118             var elem = new OrbitalElements() {
119                 semilatusrectum = 3.2*AU,
120                 eccentricity = 0.7,
121                 inclination = 1.2,
122                 ascendingNodeLongitude = 0.1,
123                 periapsisArgument = 4.3,
```

```
124                 trueAnomaly = 3.7
125             };
126             Body sun2 = (Body)sun.Clone();
127             sun2.position += new Vector3(3,2,6);
128             sun2.velocity += new Vector3(1,5,3);
129             var e1 = new Body(sun,elem);
130             var e2 = new Body(sun2,elem);
131             e2.position -= new Vector3(3,2,6);
132             e2.velocity -= new Vector3(1,5,3);
133             if (e1.position != e2.position || e1.velocity != e2.velocity) {
134                 Console.WriteLine("Parent r/v not considered");
135                 return false;
136             }
137             for (double i = 0; i < Math.PI; i += 0.2) {
138                 for (double j = 0; j < 2*Math.PI; j += 0.2) {
139                     for (double k = 0; k < 2*Math.PI; k += 0.2) {
140                         for (double l = 0; l < 2*Math.PI; l += 0.2) {
141                             for (double m = 0; l < 1; l+= 0.1) {
142                                 var earthElements = new OrbitalElements() {
143                                     semilatusrectum = 1*AU,
144                                     eccentricity = m,
145                                     inclination = i,
146                                     ascendingNodeLongitude = j,
147                                     periapsisArgument = k,
148                                     trueAnomaly = l
149                                 };
150                                 var earth = new Body(sun,earthElements){
151                                             stdGrav = 3.986004419e14,
152                                                 radius = 6.371e6,
153                                                 color = new Vector3
    (0,0.2,0.8),
154                                 };
155                                 if (m == 0) {
156                                     if (!(Math.Abs(Vector3.Magnitude
    (earth.velocity) - 3e4) < 1e3 )) {
157                                         Console.WriteLine($"{i},{j},{k},{l},
    {earth.velocity}");
158                                         return false;
159                                     } else if (!(Math.Abs(Vector3.Magnitude
    (earth.position) - 1*AU) < 1e-4 )) {
160                                         Console.WriteLine($"{i},{j},{k},{l},
    {Vector3.Magnitude(earth.position)/AU}");
161                                         return false;
162                                     }
163                                 }
164                                 var earthElements2 = new OrbitalElements
    (earth.position,earth.velocity,sun.stdGrav);
165                                 foreach (Tuple<string,double,double> t in new
    List<Tuple<string,double,double>>() {
166                                     new Tuple<string,double,double>
    ("l",earthElements.ascendingNodeLongitude,
    earthElements2.ascendingNodeLongitude),
167                                     new Tuple<string,double,double>
    ("e",earthElements.eccentricity,          earthElements2.eccentricity),
168                                     new Tuple<string,double,double>
    ("i",earthElements.inclination,           earthElements2.inclination),
169                                     new Tuple<string,double,double>
    ("w",earthElements.periapsisArgument,     earthElements2.periapsisArgument),
170                                     new Tuple<string,double,double>
    ("p",earthElements.semilatusrectum,       earthElements2.semilatusrectum),
171                                     new Tuple<string,double,double>
    ("v",earthElements.trueAnomaly,           earthElements2.trueAnomaly),
172                                 }) {
173                                     if ((t.Item2 - t.Item3)/t.Item2 > 1e-6) {
174                                         if (t.Item1 == "l" && i == 0
175                                          || (t.Item1 == "w" || t.Item1 ==
```

```
    "v") && m == 0) {
176                                                 // They are undefined, don't
    worry
177                                                 continue;
178                                             }
179                                             Console.WriteLine($"Orbital element
    test failed: {t.Item1}, {t.Item2}, {t.Item3}, {((t.Item2 - t.Item3)/
    t.Item2)*100}%");
180                                             return false;
181                                         }
182                                     }
183                                 }
184                             }
185                         }
186                     }
187                 }
188                 var elemx = new OrbitalElements() {
189                     inclination = 2*Math.PI,
190                     ascendingNodeLongitude = 7.5*Math.PI,
191                     trueAnomaly = 27*Math.PI,
192                     periapsisArgument = 3.75*Math.PI
193                 };
194                 if (
195                     elemx.inclination > 1e-10 ||
196                     (elemx.ascendingNodeLongitude - (1.5*Math.PI))/(1.5*Math.PI)
    > 1e-10 ||
197                     (elemx.trueAnomaly - Math.PI)/Math.PI > 1e-10 ||
198                     (elemx.periapsisArgument-1.75*Math.PI)/(1.75*Math.PI) > 1e-10
199                 ) {
200                     Console.WriteLine("Implicit angle readjustment failed");
201                     Console.WriteLine(elemx.trueAnomaly/Math.PI);
202                 }
203                 return true;
204             }
205             public static bool PlanetarySystemTest() {
206                 List<Body> bodies = Structures.Examples.solar_system_bodies;
207                 var sys = new PlanetarySystem(bodies);
208                 if (!bodies.SequenceEqual(((IEnumerable<Body>)sys).ToList())) {
209                     Console.WriteLine("Constructor does not add bodies");
210                     return false;
211                 }
212                 var b = Structures.Examples.solar_system_bodies[3];
213                 sys.Add(b);
214                 if (sys[sys.Count - 1] != b) {
215                     Console.WriteLine("Add() failed");
216                     return false;
217                 }
218                 var position1 = new Vector3(2,-4,12);
219                 sys = new PlanetarySystem(new List<Body>() {
220                     new Body() {stdGrav = 10},
221                     new Body() {
222                         stdGrav = 20,
223                         position = position1
224                     }
225                 });
226                 if (sys.Barycenter() != 2*position1/3) {
227                     Console.WriteLine("Barycenter 1 incorrect");
228                     return false;
229                 }
230                 sys[1].stdGrav /= 2;
231                 var position1polar = Vector3.CartesianToPolar(position1);
232                 var position2polar = new Vector3
    (position1polar.x,position1polar.y + Math.PI/3,position1polar.z);
233                 sys.Add(new Body {
234                     stdGrav = 10,
235                     position = Vector3.PolarToCartesian(position2polar)
```

```
236             });
237             double distance = Math.Sqrt(3)/3;
238             Vector3 expected_barycenter_polar = new Vector3
    (distance*position1polar.x,position1polar.y + Math.PI/6,position1polar.z);
239             if (sys.Barycenter() != Vector3.PolarToCartesian
    (expected_barycenter_polar)) {
240                 Console.WriteLine("Barycenter 2 incorrect");
241                 return false;
242             }
243             return true;
244         }
245     }
246 }
```