```
using System;
using System.Collections.Generic;
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) {
                Console.WriteLine("5*a/5 != i");
                return false;
            }
            if (a + a != 2 * a) {
                Console.WriteLine("a + a != 2 * a");
                return false;
            }
            if (-a != z - a) {
                Console.WriteLine("-a != z - a");
                return false;
            }
            if (Vector3.Magnitude(a) != 7) {
                Console.WriteLine("Incorrect Magnitude");
                return false;
            }
            if (Vector3.dot(new Vector3(1,2,0),new Vector3(-2,1,0)) != 0 || Vector3.dot(a,a) != 49) {
                Console.WriteLine("incorrect dot");
                return false;
            }
            if (Vector3.cross(new Vector3(3,-3,1), new Vector3(4,9,2)) != new Vector3(-15,-2,39)) {
                Console.WriteLine("incorrect cross");
                return false;
            }
            var a_u = new Vector3((double)2/7,(double)3/7,(double)6/7);
            if (Vector3.Unit(a) != a_u) {
                Console.WriteLine("incorrect unit");
                return false;
            }
            var exp = new Vector3(1000,0,-100);
            try {
                Console.WriteLine(Vector3.Unit(Vector3.zero));
                Console.WriteLine("Unit(zero) did not throw exception");
                return false;
            } catch (DivideByZeroException) {
            } catch (Exception) {
                Console.WriteLine("Incorrect exception");
                return false;
            }
            if (Vector3.PolarToCartesian(Vector3.CartesianToPolar(a)) != a) {
                var b = Vector3.PolarToCartesian(Vector3.CartesianToPolar(a));
                Console.WriteLine((a.x - b.x)/a.x);
                Console.WriteLine("Cartesian-Polar conversions failed");
                return false;
            }
            Vector3 c = null;
            Vector3 d = null;
            if (a == c || c != d) {
                Console.WriteLine("Null checks incorrect");
                return false;
            }
            return true;
        }

        public static bool BodyTest() {
            var sun = new Body {
                        stdGrav = 1.3271440019e20,
                        radius = 6.95e8
            };
            var elem = new OrbitalElements() {
                semilatusrectum = 3.2*AU,
                eccentricity = 0.7,
                inclination = 1.2,
                ascendingNodeLongitude = 0.1,
                periapsisArgument = 4.3,
                trueAnomaly = 3.7
            };
            Body sun2 = (Body)sun.Clone();
            sun2.position += new Vector3(3,2,6);
            sun2.velocity += new Vector3(1,5,3);
            var e1 = new Body(sun,elem);
            var e2 = new Body(sun2,elem);
            e2.position -= new Vector3(3,2,6);
            e2.velocity -= new Vector3(1,5,3);
            if (e1.position != e2.position || e1.velocity != e2.velocity) {
                Console.WriteLine("Parent r/v not considered");
                return false;
            }
            for (double i = 0; i < Math.PI; i += 0.2) {
                for (double j = 0; j < 2*Math.PI; j += 0.2) {
                    for (double k = 0; k < 2*Math.PI; k += 0.2) {
                        for (double l = 0; l < 2*Math.PI; l += 0.2) {
                            for (double m = 0; l < 1; l+= 0.1) {
                                var earthElements = new OrbitalElements() {
                                    semilatusrectum = 1*AU,
                                    eccentricity = m,
                                    inclination = i,
                                    ascendingNodeLongitude = j,
                                    periapsisArgument = k,
                                    trueAnomaly = l
                                };
                                var earth = new Body(sun,earthElements){
                                            stdGrav = 3.986004419e14,
                                                radius = 6.371e6,
                                                color = new Vector3(0,0.2,0.8),
                                };
                                if (m == 0) {
                                    if (!(Math.Abs(Vector3.Magnitude(earth.velocity) - 3e4) < 1e3 )) {
                                        Console.WriteLine($"{i},{j},{k},{l},{earth.velocity}");
                                        return false;
                                    } else if (!(Math.Abs(Vector3.Magnitude(earth.position) - 1*AU) < 1e-4 )) {
                                        Console.WriteLine($"{i},{j},{k},{l},{Vector3.Magnitude(earth.position)/AU}");
                                        return false;
                                    }
                                }
                                var earthElements2 = new OrbitalElements(earth.position,earth.velocity,sun.stdGrav);
                                foreach (Tuple<string,double,double> t in new List<Tuple<string,double,double>>() {
                                    new Tuple<string,double,double>("l",earthElements.ascendingNodeLongitude, earthElements2.ascendingNodeLongitude),
                                    new Tuple<string,double,double>("e",earthElements.eccentricity,            earthElements2.eccentricity),
                                    new Tuple<string,double,double>("i",earthElements.inclination,             earthElements2.inclination),
                                    new Tuple<string,double,double>("w",earthElements.periapsisArgument,      earthElements2.periapsisArgument),
                                    new Tuple<string,double,double>("p",earthElements.semilatusrectum,        earthElements2.semilatusrectum),
                                    new Tuple<string,double,double>("v",earthElements.trueAnomaly,            earthElements2.trueAnomaly),
                                }) {
                                    if ((t.Item2 - t.Item3)/t.Item2 > 1e-6) {
                                        if (t.Item1 == "l" && i == 0
                                         || (t.Item1 == "w" || t.Item1 == "v") && m == 0) {
                                            // They are undefined, don't worry
                                            continue;
                                        }
                                        Console.WriteLine($"Orbital element test failed: {t.Item1}, {t.Item2}, {t.Item3}, {((t.Item2 - t.Item3)/t.Item2)*100}%");
                                        return false;
                                    }
                                }
                            }
                        }
                    }
                }
            }
            var elemx = new OrbitalElements() {
                inclination = 2*Math.PI,
                ascendingNodeLongitude = 7.5*Math.PI,
                trueAnomaly = 27*Math.PI,
                periapsisArgument = 3.75*Math.PI
            };
            if (
                elemx.inclination > 1e-10 ||
                (elemx.ascendingNodeLongitude - (1.5*Math.PI))/(1.5*Math.PI) > 1e-10 ||
                (elemx.trueAnomaly - Math.PI)/Math.PI > 1e-10 ||
                (elemx.periapsisArgument-1.75*Math.PI)/(1.75*Math.PI) > 1e-10
            ) {
                Console.WriteLine("Implicit angle readjustment failed");
                Console.WriteLine(elemx.trueAnomaly/Math.PI);
            }
            return true;
        }
        public static bool PlanetarySystemTest() {
            List<Body> bodies = Structures.Examples.solar_system_bodies;
            var sys = new PlanetarySystem(bodies);
            if ((IEnumerator<Body>)bodies.GetEnumerator() != sys.GetEnumerator()) {
                Console.WriteLine("Constructor does not add bodies");
                return false;
            }
            var b = Structures.Examples.solar_system_bodies[3];
            sys.Add(b);
            if (sys[sys.Count - 1] != b) {
                Console.WriteLine("Add() failed");
                return false;
            }
            var position1 = new Vector3(2,-4,12);
            sys = new PlanetarySystem(new List<Body>() {
                new Body() {stdGrav = 10},
                new Body() {
                    stdGrav = 20,
                    position = position1
                }
            });
            if (sys.Barycenter() != 2*position1/3) {
                Console.WriteLine("Barycenter 1 incorrect");
                return false;
            }
            sys[1].stdGrav /= 2;
            var position1polar = Vector3.CartesianToPolar(position1);
            var position2polar = new Vector3(position1polar.x,position1polar.y + Math.PI/3,position1polar.z);
            sys.Add(new Body {
                stdGrav = 10,
                position = Vector3.PolarToCartesian(position2polar)
            });
            double distance = Math.Sqrt(3)/3;
            Vector3 expected_barycenter_polar = new Vector3
(distance*position1polar.x,position1polar.y + Math.PI/6,position1polar.z);
            if (sys.Barycenter() != Vector3.PolarToCartesian
(expected_barycenter_polar)) {
                Console.WriteLine("Barycenter 2 incorrect");
                return false;
            }
            return true;
        }
    }
}
```