**Functional testing**

**Contents of Program.cs file (Equation):**

using Library;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace umop7o8iFunctionalTesting

{

class Program

{

static void Main(string[] args)

{

var a = double.Parse(Console.ReadLine());

var b = double.Parse(Console.ReadLine());

var c = double.Parse(Console.ReadLine());

var result = QuadricEquation.Solve(a, b, c);

Console.WriteLine(result[0]);

Console.WriteLine(result[1]);

Console.ReadKey();

}

}

}

**Contents of QuadricEquation.cs file (Library):**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Library

{

public class QuadricEquation

{

public static double[] Solve (double a, double b, double c)

{

if (a == 0)

return new double[] { -c / b };

var squareOfDisc = Math.Sqrt(b \* b - 4 \* a \* c);

if (double.IsNaN(squareOfDisc))

return new double[0];

if (squareOfDisc == 0)

return new double[] { -b / (2 \* a) };

return new[]

{

(- b + squareOfDisc)/(2\*a),

(- b - squareOfDisc)/(2\*a)

};

}

}

}

**Contents of UnitTest.cs file (TestProject):**

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Library;

namespace TestProject

{

[TestClass]

public class UnitTest

{

void Test(double a, double b, double c, int length, double x1, double x2)

{

var result = QuadricEquation.Solve(a, b, c);

Assert.AreEqual(length, result.Length);

if (length > 0)

Assert.AreEqual(result[0], x1);

if (length > 1)

Assert.AreEqual(result[1], x2);

}

[TestMethod]

public void OrdinaryCase()

{

Test(2, -4, -6, 2, 3, -1);

}

[TestMethod]

public void NegativeDiscriminant()

{

Test(-2, -4, -6, 0, 0, 0);

}

[TestMethod]

public void ZeroA()

{

Test(0, 1, 1, 1, -1, 0);

}

[TestMethod]

public void FunctionalTest()

{

var rnd = new Random(1);

for (int i = 0; i < 100; i++)

{

var a = rnd.NextDouble() \* 10;

var b = rnd.NextDouble() \* 10;

var c = rnd.NextDouble() \* 10;

var solution = QuadricEquation.Solve(a, b, c);

for (int j = 0; j < solution.Length; j++)

{

Assert.AreEqual(0, a \* solution[j] \* solution[j] + b \* solution[j] + c, 10e-6);

}

}

}

}

}