Постановка задачи

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
   - TDD - фреймворк (не менее 3 тестов).
   - BDD - фреймворк (не менее 3 тестов).

Код программы

Main.py

```
class LevenshteinDistance:
    def dist(self, a, b):
        def rec(i, j):
            if i == 0 or j == 0:
                return max(i, j)
            elif a[i - 1] == b[j - 1]:
                return rec(i - 1, j - 1)
            else:
                return 1 + min(rec(i, j - 1), rec(i - 1, j), rec(i - 1, j -
1))

        return rec(len(a), len(b))


if __name__ == "__main__":
    str1 = input("Введите первое слово:")
    str2 = input("Введите второе слово:")

    levenshtein = LevenshteinDistance()
    lev = levenshtein.dist(str1, str2)
    print("Расстояние Левенштейна:" + str(lev))
```

test.py

```
# TDD - фреймворк (не менее 3 тестов).

import unittest
from main import LevenshteinDistance

class TestLevenshteinDistance(unittest.TestCase):
    def test_distance_equal_strings(self):
        levenshtein = LevenshteinDistance()
        self.assertEqual(levenshtein.dist("kitten", "kitten"), 0)

    def test_distance_different_strings(self):
        levenshtein = LevenshteinDistance()
        self.assertEqual(levenshtein.dist("kitten", "sitting"), 3)

    def test_distance_empty_string(self):
        levenshtein = LevenshteinDistance()
        self.assertEqual(levenshtein.dist("", "abc"), 3)
```

```python
if __name__ == '__main__':
    unittest.main()
```

test2.py

```python
from main import LevenshteinDistance
# Тесты с использованием pytest (BDD)
# BDD - фреймворк (не менее 3 тестов).
import pytest

@pytest.fixture
def levenshtein_distance():
    return LevenshteinDistance()

def test_distance_equal_strings(levenshtein_distance):
    assert levenshtein_distance.dist("kitten", "kitten") == 0

def test_distance_different_strings(levenshtein_distance):
    assert levenshtein_distance.dist("kitten", "sitting") == 3

def test_distance_empty_string(levenshtein_distance):
    assert levenshtein_distance.dist("", "abc") == 3

if __name__ == '__main__':
    pytest.main()
```

Код программы на C#

Program.cs

```csharp
using System;

class LevenshteinDistance
{
   public int Dist(string a, string b)
   {
      int Rec(int i, int j)
      {
         if (i == 0 || j == 0)
         {
            return Math.Max(i, j);
         }
         else if (a[i - 1] == b[j - 1])
         {
            return Rec(i - 1, j - 1);
         }
         else
         {
            return 1 + Math.Min(Rec(i, j - 1), Math.Min(Rec(i - 1, j), Rec(i - 1, j - 1)));
         }
      }

      return Rec(a.Length, b.Length);
   }

   static void Main()
   {
      Console.Write("Введите первое слово: ");
```

```
        string str1 = Console.ReadLine();

        Console.Write("Введите второе слово: ");
        string str2 = Console.ReadLine();

        LevenshteinDistance levenshtein = new LevenshteinDistance();
        int lev = levenshtein.Dist(str1, str2);
        Console.WriteLine("Расстояние Левенштейна: " + lev);
    }
}
```

Test

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

[TestClass]
public class TestLevenshteinDistance
{
    [TestMethod]
    public void TestDistanceEqualStrings()
    {
        // Arrange
        LevenshteinDistance levenshtein = new LevenshteinDistance();

        // Act
        int result = levenshtein.Dist("kitten", "kitten");

        // Assert
        Assert.AreEqual(0, result);
    }

    [TestMethod]
    public void TestDistanceDifferentStrings()
    {
        // Arrange
        LevenshteinDistance levenshtein = new LevenshteinDistance();

        // Act
        int result = levenshtein.Dist("kitten", "sitting");

        // Assert
        Assert.AreEqual(3, result);
    }

    [TestMethod]
    public void TestDistanceEmptyString()
    {
        // Arrange
        LevenshteinDistance levenshtein = new LevenshteinDistance();

        // Act
        int result = levenshtein.Dist("", "abc");

        // Assert
        Assert.AreEqual(3, result);
    }
}
```

Test2

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using NUnit.Framework;

[TestClass]
public class TestLevenshteinDistanceMSTest
{
    [TestMethod]
    public void TestDistanceEqualStrings()
    {
        LevenshteinDistance levenshtein = new LevenshteinDistance();
        Assert.AreEqual(0, levenshtein.Dist("kitten", "kitten"));
    }

    [TestMethod]
    public void TestDistanceDifferentStrings()
    {
        LevenshteinDistance levenshtein = new LevenshteinDistance();
        Assert.AreEqual(3, levenshtein.Dist("kitten", "sitting"));
    }

    [TestMethod]
    public void TestDistanceEmptyString()
    {
        LevenshteinDistance levenshtein = new LevenshteinDistance();
        Assert.AreEqual(3, levenshtein.Dist("", "abc"));
    }
}

[TestFixture]
public class TestLevenshteinDistanceNUnit
{
    [Test]
    public void TestDistanceEqualStrings()
    {
        LevenshteinDistance levenshtein = new LevenshteinDistance();
        NUnit.Framework.Assert.AreEqual(0, levenshtein.Dist("kitten", "kitten"));
    }

    [Test]
    public void TestDistanceDifferentStrings()
    {
        LevenshteinDistance levenshtein = new LevenshteinDistance();
        NUnit.Framework.Assert.AreEqual(3, levenshtein.Dist("kitten", "sitting"));
    }

    [Test]
    public void TestDistanceEmptyString()
    {
        LevenshteinDistance levenshtein = new LevenshteinDistance();
        NUnit.Framework.Assert.AreEqual(3, levenshtein.Dist("", "abc"));
    }
}
```

Анализ результатов

```
Ran 3 tests in 0.006s

OK
```