﻿using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using SecurityLibrary;

using System.Collections.Generic;

namespace SecurityPackageTest

{

[TestClass]

public class HillCipherTest

{

List<int> key = new List<int>() { 3, 2, 8, 5 };

List<int> plain = new List<int>() { 15, 0, 24, 12, 14, 17, 4, 12, 14, 13, 4, 24 };

List<int> cipher = new List<int>() { 19, 16, 18, 18, 24, 15, 10, 14, 16, 21, 8, 22 };

string mainPlain = "paymoremoney";

string mainCipher = "tqssypkoqviw".ToUpper();

string mainPlainError = "lkdi";

string mainCipherError = "SDEK".ToUpper();

List<int> mainPlainError2 = new List<int>() { 11, 10, 3, 8 };

List<int> mainCipherError2 = new List<int>() { 18, 3, 4, 10 };

List<int> keyError = new List<int>() { 11, 10, 3, 8 };

string mainKey = "dcif";

List<int> key3 = new List<int> { 17, 17, 5, 21, 18, 21, 2, 2, 19 };

List<int> cipher3 = new List<int> { 11, 13, 18, 7, 3, 11, 4, 22, 12, 19, 17, 22 };

string keyS3 = "rrfvsvcct";

string cipherS3 = "lnshdlewmtrw".ToUpper();

string mainPlain3 = "fvcfcqtob";

string mainCipher3 = "hgrzeudvq".ToUpper();

string mainKey3 = "bkaaubcpc";

List<int> plain4 = new List<int> { 5, 21, 2, 5, 2, 16, 19, 14, 1 };

List<int> cipher4 = new List<int> { 7, 6, 17, 25, 4, 20, 3, 21, 16 };

List<int> key4 = new List<int> { 1, 10, 0, 0, 20, 1, 2, 15, 2 };

string newPlain = "thegoldisburiedinorono";

string newCipher = "gzscxnvcdjzxeovcrclsrc".ToUpper();

string newKey = "frep";

#region integer test cases

[TestMethod]

public void HillCipherTestEnc2()

{

HillCipher algorithm = new HillCipher();

List<int> cipher2 = algorithm.Encrypt(plain, key);

for (int i = 0; i < cipher.Count; i++)

{

Assert.AreEqual(cipher[i], cipher2[i]);

}

}

[TestMethod]

public void HillCipherTestDec2()

{

HillCipher algorithm = new HillCipher();

List<int> plain2 = algorithm.Decrypt(cipher, key);

for (int i = 0; i < plain.Count; i++)

{

Assert.AreEqual(plain[i], plain2[i]);

}

}

[TestMethod]

public void HillCipherTest2By2Analysis2()

{

HillCipher algorithm = new HillCipher();

List<int> key2 = algorithm.Analyse(plain, cipher);

for (int i = 0; i < key.Count; i++)

{

Assert.AreEqual(key[i], key2[i]);

}

}

[TestMethod]

public void HillCipherTestEnc4()

{

HillCipher algorithm = new HillCipher();

List<int> cipher2 = algorithm.Encrypt(plain, key3);

for (int i = 0; i < cipher3.Count; i++)

{

Assert.AreEqual(cipher3[i], cipher2[i]);

}

}

[TestMethod]

public void HillCipherTestDec4()

{

HillCipher algorithm = new HillCipher();

List<int> plain2 = algorithm.Decrypt(cipher3, key3);

for (int i = 0; i < plain.Count; i++)

{

Assert.AreEqual(plain[i], plain2[i]);

}

}

[TestMethod]

public void HillCipherTestEnc6()

{

HillCipher algorithm = new HillCipher();

List<int> cipher2 = algorithm.Encrypt(plain4, key4);

for (int i = 0; i < cipher4.Count; i++)

{

Assert.AreEqual(cipher4[i], cipher2[i]);

}

}

[TestMethod]

public void HillCipherTestDec6()

{

HillCipher algorithm = new HillCipher();

List<int> plain2 = algorithm.Decrypt(cipher4, key4);

for (int i = 0; i < plain4.Count; i++)

{

Assert.AreEqual(plain4[i], plain2[i]);

}

}

[TestMethod]

public void HillCipherTest3By3Analysis2()

{

HillCipher algorithm = new HillCipher();

List<int> key2 = algorithm.Analyse3By3Key(plain4, cipher4);

for (int i = 0; i < key4.Count; i++)

{

Assert.AreEqual(key4[i], key2[i]);

}

}

#endregion

#region error test cases (key with no inverse)

[TestMethod]

[ExpectedException(typeof(InvalidAnlysisException))]

public void HillCipherError2()

{

HillCipher algorithm = new HillCipher();

List<int> key2 = algorithm.Analyse(mainPlainError2, mainCipherError2);

}

// Decrypt with invalid key

[TestMethod]

[ExpectedException(typeof(System.Exception), AllowDerivedTypes = true)]

public void HillCipherError3()

{

HillCipher algorithm = new HillCipher();

List<int> key2 = algorithm.Decrypt(plain, keyError);

}

#endregion

}

}