МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

ЗВІТ
з лабораторної роботи № 9
«Модулярна (модульна) арифметика та її застосування. 121 – Інженерія програмного забезпечення. 2021–2022 н.р.»

Дисципліна «Основи програмування»

Виконала :
студентка 1-го курсу, групи КП-11,
спеціальності 121 – Інженерія
програмного забезпечення
Кирильчук Олександра Артурівна

Перевірив:
к. т. н, старший викладач
Хайдуров Владислав
Володимирович

Київ – 2021

Завдання до лабораторної роботи для виконання

1. Ознайомитись з теоретичними відомостями до лабораторної роботи загалом. Ознайомитись з основними принципами модульної арифметики.
2. Реалізувати програмно шифрування та розшифрування будь-якої текстової інформації алгоритмом Цезаря. Ключ також обрати / згенерувати самостійно.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Caesar
{
    class Program
    {
        public static char encription(char en, int key)
        {
            if (!char.IsLetter(en)) //Показывает, относится ли указанный символ Юникода к категории букв Юникода
            {
                return en;
            }
            char d = char.IsUpper(en) ? 'A' : 'a'; //Показывает, относится ли указанный символ Юникода к категории букв верхнего регистра
            return (char)((((en + key) - d) % 26) + d);
        }
        public static string Encription(string InputText, int Key)
        {
            string Encripted = string.Empty;
            foreach (char en in InputText)
                Encripted += encription(en, Key);
            return Encripted;
        }
        public static string ReEncription(string InputText, int Key)
        {
            return Encription(InputText, 26 - Key);
        }
        static void Main(string[] args)
        {
            Console.WriteLine("Input Text :");
            string InputText = Console.ReadLine();
            Console.WriteLine();
            Console.WriteLine("Input Key : ");
            int Key = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            Console.WriteLine("Encrypted Text : ");
            string EncriptedText = Encription(InputText, Key);
            Console.WriteLine(EncriptedText);
            Console.WriteLine();
            Console.WriteLine("Re-Ecrypted Text : ");
            string ReEcryptedText = ReEncription(EncriptedText, Key);
            Console.WriteLine(ReEcryptedText);
            Console.WriteLine();
            Console.ReadKey();
        }
    }
}
```

```
Input Text :
Breathtaking

Input Key :
5

Encrypted Text :
Gwjfymyfpnsl

Re-Ecrypted Text :
Breathtaking
```

3. Ознайомитись з алгоритмом Діффі-Хеллмана для обміну ключами. Згенерувати дані для обміну. Написати відповідну програму для обміну ключами. Вхідні дані для обміну обрати / згенерувати самостійно.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

class DiffiHellman
{
    static void Main()
    {
        Console.WriteLine("p = " + Alice.p);
        Console.WriteLine("g = " + Alice.g);
        Console.WriteLine("A = "+ Alice.A);
        Console.WriteLine("Alice K = " + Alice.K);
        Console.WriteLine("B = " + Bob.B);
        Console.WriteLine("Bob K = " + Bob.K);
        Console.ReadKey();
    }
}
class Alice
{
    public const int p = 23;
    public const int g = 5;
    private const int a = 4;
    public static int A = (int)Math.Pow(g, a) % p;
    public static int K = (int)Math.Pow(Bob.B, a) % p;
}
class Bob
{
    private const int b = 3;
    public static int B = (int)Math.Pow(Alice.g, b) % Alice.p;
    public static int K = (int)Math.Pow(Alice.A, b) % Alice.p;
}
```

```
p = 23
g = 5
A = 4
Alice K = 18
B = 10
Bob K = 18
```

4. Ознайомитись з алгоритмом піднесення до степені за принципом схеми Горнера(1) (швидке піднесення до степеня), а також з алгоритмом знаходження оберненого числа за модулем (розширений алгоритм Евкліда(2)).

(1)

```
using System;
using System.Numerics;
namespace ads
{
    class Program
    {
        static BigInteger Hor(BigInteger x, int e, int y)
        {
            string s = Convert.ToString(e, 2);
            BigInteger res = 1, tem = x % y;
            for (int i = 0; i < s.Length; i++)
            {
                if (s[^(i + 1)] == '1')
                    res = res * tem % y;
                tem = tem * tem % y;
            }
            return res;
        }

        static void Main(string[] args)
        {
            BigInteger x = 61;
            int y = 23, e = 15;
            Console.WriteLine(Hor(x, e, y));
            Console.WriteLine(x);
            Console.WriteLine(y);
            Console.WriteLine(e);
        }
    }
}
```

```
21
61
23
15
```

(2)

```
using System;
namespace ads
{
    class Program
    {
        static int Evc(int a, int b, out int x, out int y)
        {
            if (b < a)
            {
                var t = a;
                a = b;
                b = t;
            }
            if (a == 0)
            {
                x = 0;
                y = 1;
                return b;
            }
            int evc = Evc(b % a, a, out x, out y);
            int newY = x;
            int newX = y - (b / a) * x;
            x = newX;
            y = newY;
            return evc;
        }
        static void Main(string[] args)
```

```
        {
            int x;
            int y;
            Console.WriteLine(Evc(13, 396, out x, out y));
            Console.WriteLine(x);
            Console.WriteLine(y);
        }
    }
}
```

```
1
61
-2
```

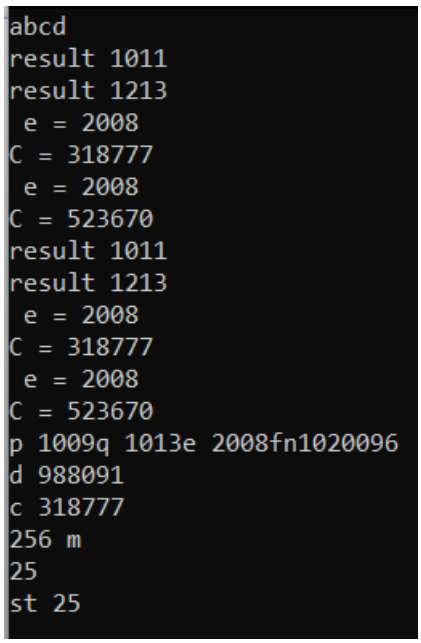
5. Ознайомитись з криптосистемою RSA та реалізувати його з використанням алгоритму швидкого піднесення до степеня, а також розширеного алгоритму Евкліда. Для демонстрації роботи алгоритму взяти будь-які текстові дані. Згенерувати константи (усі більші за 100000) для даного алгоритму та виконати шифрування та дешифрування текстових даних. У процесі шифрування брати повідомлення M по кілька символів (шифрування виконується для кожних k символів, k > 1).

```
using System;
using System.Collections.Generic;
using System.Numerics;
using System.Linq;

namespace RSA
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = Console.ReadLine();
            B.Encription(s);
            A.ReEncription(B.Encription(s));
            Console.WriteLine();
            Console.ReadLine();
        }
    }
    class A
    {
        Random rnd = new Random();
        private static BigInteger p = Prost(1000), q = Prost(p + 1), x, y;
        public static BigInteger n = p * q;
        private static BigInteger fn = Eyler(n);
        public static BigInteger e = 2008;
        private static BigInteger d = Evc(e, fn, out x, out y);
        private static BigInteger Prost(BigInteger M)
        {
            for (BigInteger i = M; i < 5000; i++)
            {
                 if(isSimple(i))
                 {
                    return i;
                    break;
                 }
            }
            return 1;
        }

        private static bool isSimple(BigInteger N)
        {
```

```
    bool tf = false;
    for (BigInteger i = 2; i < (int)(N / 2); i++)
    {
        if (N % i == 0)
        {
            tf = false;
            break;
        }
        else
        {
            tf = true;
        }
    }
    return tf;
}
private static BigInteger E (BigInteger fn)
{
    for (BigInteger i = fn - 1; i < fn; i--)
    {
        if (IsCoprime(i, fn) == true)
        {
            return i;
            break;
        }
    }
    return 0;
}
public static bool IsCoprime(BigInteger num1, BigInteger num2)
{
    if (num1 == num2)
    {
        return num1 == 1;
    }
    else
    {
        if (num1 > num2)
        {
            return IsCoprime(num1 - num2, num2);
        }
        else
        {
            return IsCoprime(num2 - num1, num1);
        }
    }
}

public static void ReEncription(List<BigInteger> C)
{
    Console.WriteLine("p " + p + "q " + q + "e " + e + "fn" + fn);
    Console.WriteLine("d " + d);
    string m;
    string Result = null;
    int st;
    foreach(int c in C)
    {
        Console.WriteLine("c "+ c);
        m = B.Hor(c, d, n).ToString();

        Console.WriteLine(m + " m");
        for (int i = 0; i < m.Length; i+=2)
        {
            Console.WriteLine(m.Substring(i, 2));
            st = Convert.ToInt32(m.Substring(i, 2));
            Console.WriteLine("st " + st);
            Result += B.let[st - 10];
```

```
                }


            }
            Console.WriteLine(Result);
        }
        public static BigInteger Eyler(BigInteger n)
        {
            BigInteger res = n, en = (BigInteger)(Math.Exp(BigInteger.Log(n) / 2) + 1);
            for (int i = 2; i <= en; i++)
                if ((n % i) == 0)
                {
                    while ((n % i) == 0)
                        n /= i;
                    res -= (res / i);
                }
            if (n > 1) res -= (res / n);
            return res;
        }
        public static BigInteger Evc(BigInteger a, BigInteger b, out BigInteger x, out
BigInteger y)
        {
            if (b < a)
            {
                var t = a;
                a = b;
                b = t;
            }
            if (a == 0)
            {
                x = 0;
                y = 1;
                return b;
            }
            BigInteger evc = Evc(b % a, a, out x, out y);
            BigInteger newY = x;
            BigInteger newX = y - (b / a) * x;
            x = newX;
            y = newY;
            return x + b;
        }

    }
    class B
    {
        public int M = 1;
        public static char[] let = new char[] {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',
'i', 'j', 'k', 'l', 'm', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',' '
};
        public static BigInteger[] num = new BigInteger[] { 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37 };
        public static List<BigInteger> Encription(string s)
        {
            BigInteger m = 0, n = 0, g =0;
            List<BigInteger> M = new List<BigInteger>();
            List<BigInteger> C = new List<BigInteger>();
            BigInteger result = 0;
            for (int i = 0; i < s.Length - 1; i += 2)
            {
                for (int j = 0; j < let.Length; j++)
                {
                    if (s.Length % 2 != 0 && i == 0)
                    {

                        if (s[i] == let[j])
```

```
                        m = num[j];
                    if (s[i + 1] == let[j])
                        n = num[j];
                    if(s[i+2] == let[j])
                    {
                        g = num[j];
                    }
                    result = m * 10000 + n * 100 + g;
                }
                else
                {

                    if (s[i] == let[j])
                        m = num[j];
                    if (s[i + 1] == let[j])
                        n = num[j];
                    result = m * 100 + n;
                }
            }
            Console.WriteLine("result " + result);
            M.Add(result);
        }
        for (int i = 0; i < M.Count; i++)
        {
            Console.WriteLine(" e = " + A.e);
            C.Add(Hor(M[i], A.e, A.n));
            Console.WriteLine("C = "+C[i]);
        }
        return C;
    }
    public static BigInteger Hor(BigInteger x, BigInteger e, BigInteger y)
    {
        string s = Convert.ToString((int)e, 2);
        BigInteger res = 1, tem = x % y;
        for (int i = 0; i < s.Length; i++)
        {
            if (s[^(i + 1)] == '1')
                res = res * tem % y;
            tem = tem * tem % y;
        }
        return res;

    }
}
}
```



//Зашифровує правильно, але десь помилка

6. Усі результати виконання лабораторної роботи занести до звіту.