```
// Labor 1 - Feladat 2
#include "stdafx.h"
#include <iostream>
#include <fstream>
using namespace std;
int crypt(char *inf, char *outf);
int encrypt(char *inf, char *outf);
int main() {
	crypt("kep.jpg", "kript.jpg");
	encrypt("kript.jpg", "ujkep.jpg");
	cout << "succesful copy..." << endl;
	return 0;
}
int crypt(char *inf, char *outf)
{
	ifstream in(inf, ifstream::binary);
	if (!in.is_open())
	{
		cout << "Error opening in_file\n";
		return 0;
	}
	ofstream out(outf, ifstream::binary);
	if (!out.is_open())
	{
		cout << "Error opening out_file\n";
		return 0;
	}
	int tempk, k1, k2, k, key;
	cout << "enter the size of byte block: ";
	cin >> tempk;
	cout << "enter the key: ";
```

```
cin >> key;
in.seekg(0, ios::end);
int fsize = in.tellg();
in.seekg(0, ios::beg);

cout << "file size: " << fsize << endl;
k2 = fsize / tempk;
k1 = fsize % tempk;

char *bytes = new char[tempk + 1];
char *cbytes = new char[tempk + 1];
for (int i = 0; i < k2 + 1; ++i)
{
	if (i == k2)
		k = k1;
	else
		k = tempk;
	in.read(bytes, k);

	if (i == k2)
	{
		for (int i = 0; i < k; i++)
		{
			cbytes[i] = (bytes[i] + key) % 256;
		}
	}
	else
	{
		for (int i = 0; i < tempk; i++)
		{
			cbytes[i] = (bytes[i] + key) % 256;
```

```
                        }
                }
                out.write(cbytes, k);
        }
        out.close();
        in.close();
        return 0;
}
int encrypt(char *inf, char *outf)
{
        ifstream in(inf, ifstream::binary);

        if (!in.is_open())
        {
                cout << "Error opening in_file\n";
                return 0;
        }

        ofstream out(outf, ifstream::binary);

        if (!out.is_open())
        {
                cout << "Error opening out_file\n";
                return 0;
        }

        int tempk, k1, k2, k, key;
        cout << "enter the size of byte block: ";
        cin >> tempk;
        cout << "enter the key: ";
        cin >> key;
```

```
in.seekg(0, ios::end);
int fsize = in.tellg();
in.seekg(0, ios::beg);

cout << "file size: " << fsize << endl;
k2 = fsize / tempk;
k1 = fsize % tempk;

char *cbytes = new char[tempk + 1];
char *bytes = new char[tempk + 1];
for (int i = 0; i < k2 + 1; ++i)
{
	if (i == k2)
		k = k1;
	else
		k = tempk;
	in.read(cbytes, k);
	if (i == k2)
	{
		for (int i = 0; i < k; i++)
		{
			bytes[i] = (cbytes[i] + 256 - key) % 256;
		}
	}
	else
	{
		for (int i = 0; i < tempk; i++)
		{
			bytes[i] = (cbytes[i] + 256 - key) % 256;
		}
	}
```

```
                out.write(bytes, k);
            }
            out.close();
            in.close();
            return 0;
}
```

```
// Labor 1 - Feladat 2 Ilyes Hunor Levente
#include <iostream>
#include <fstream>
using namespace std;
int Caesar(char *inf, char *outf);
int main() {
    Caesar("cryptCaesar.txt", "DeCrypt.txt");
    cout << "succesful done..." << endl;
    return 0;
}

int Caesar(char *inf, char *outf)
{
    ifstream in(inf);
    if (!in.is_open()) {
        cout << "Error opening in_file\n";
        return 0;
    }

    ofstream out(outf);
    if (!out.is_open()) {
        cout << "Error opening out_file\n";
        return 0;
    }
```

```
in.seekg(0, ios::end);
int fsize = in.tellg();
in.seekg(0, ios::beg);

string text = "";

for (int i = 0; i < fsize; ++i)
{
   in >> noskipws >> text[i];
}

for (int j = 0; j < 26; j++)
{
   char *C = new char[fsize];
   int key = j;

   for (int i = 0; i < fsize; ++i)
   {
      if (text[i] >= 'A' && text[i] <= 'Z')
      {
         C[i] = ( ( ( text[i] - 'A' ) + 26 - key ) % 26 ) + 'A';
      }
      else
      {
         C[i] = text[i];
      }
   }
```

```
        for (int i = 0; i < fsize; i++)
        {
            out << C[i];
        }
        out << endl << endl;
    }

    in.close();
    out.close();

    return 0;
}
```

//Labor 2 - Feladat 3 Ilyes Hunor Levente

```
#include <iostream>
#include <fstream>
using namespace std;
int Affine(char *inf, char *out);

int main() {
    Affine("cryptAffine.txt", "kep.jpg");
    return 0;
}

int Affine(char *inf, char *outf)
{
    ifstream in(inf, ifstream::binary);
    if (!in.is_open()) {
        cout << "Error opening in_file\n";
        return 0;
    }
```

```
ofstream out(outf, ifstream::binary);
if (!out.is_open()) {
   cout << "Error opening out_file\n";
   return 0; }
unsigned char vbyte1, vbyte2, byte1, byte2;
int a, b;
in.read( (char *) &byte1, 1 );
in.read( (char *) &byte2, 1 );

for( a = 1; a < 256; a += 2 )
{
   for ( b = 0; b < 256; b += 1 )
   {
      vbyte1 = ( (byte1 + 256 - b) * a ) % 256;
      vbyte2 = ( (byte2 + 256 - b) * a ) % 256;
      if( vbyte1 == 0xFF && vbyte2 == 0xD8 )
      {
         break;
      }
   }

   if( vbyte1 == 0xFF && vbyte2 == 0xD8 )
   {
      break;
   }
}
in.seekg(0, ios::beg);
while (true)
{
   in.read( (char *) &byte1, 1);
   vbyte1 = ( (byte1 + 256 - b) * a ) % 256;
```

```
        if( in.eof() )
        {
            break;
        }
        out.write( (char *) &vbyte1, 1 );
    }
    out.close();
    in.close();
    return 0;
}
```

```
//Labor 2 - Feladat 4 Ilyes Hunor Levente
#include <iostream>
#include <fstream>
using namespace std;
int Affine(char *inf, char *out);

int main() {
    Affine("crypt.txt", "deCrypt.txt");
    return 0;
}
int Affine(char *inf, char *outf)
{
    ifstream in(inf);
    if (!in.is_open()) {
        cout << "Error opening in_file\n";
        return 0;
    }
    ofstream out(outf);
    if (!out.is_open()) {
        cout << "Error opening out_file\n";
        return 0; }
```

```
in.seekg(0, ios::end);
int fsize = in.tellg();
in.seekg(0, ios::beg);
int a,b,a1  = 0;
char *C     = new char[fsize];
char *IN    = new char[fsize];

for (int i = 0; i < fsize; ++i)
{
   in >> noskipws >>IN[i];
}
for (a = 1; a < 26; a+=2)
{
   if ((('S' - 'K') * a) % 26 == ('X' - 'J'))
   {
      break;
   }
}
for (b = 0; b < 26; b++)
{
   if ((('S' - 'A') * a + b ) % 26  == ('X' - 'A'))
   {
      break;
   }
}
for(a1=1; a1 < 26; a1+=2)
{
   if((a * a1) % 26 == 1 )
   {
      break; }}
```

```
    cout << a << endl;
    cout << b << endl;
    cout << a1 <<endl;

    for (int i = 0; i < fsize; ++i)
    {
       if (IN[i] >= 'A' && IN[i] <= 'Z')
       {
          C[i] = ( ( (IN[i] - 'A' + 26 - b ) * a1) % 26 ) + 'A';
       }
       else
       {
          C[i] = IN[i];
       }
    }

    for (int i = 0; i < fsize; i++)
    {
       out << C[i];
    }
     in.close();
     out.close();
}
```

//Labor 3 - Feladat 4 Ilyes Hunor Levente

```
#include <NTL/matrix.h>
#include <NTL/mat_ZZ.h>
#include <time.h>
#include <iomanip>
#include <fstream>
#include <iostream>
```

```
using namespace std;
using namespace NTL;

void my_writeM(Mat<ZZ>key) {
	for (int i = 0; i < key.NumRows(); ++i) {
		for (int j = 0; j < key.NumCols(); ++j)
			cout << setw(6) << key[i][j]%256;
		cout << endl;
	}
}
void key_det(Mat<ZZ> &invKey,int n) {

	Mat<ZZ> key, m, c, mInv, adjM, adjKey;

	m.SetDims(n,n);
	c.SetDims(n,n);

	m[0][0]=0x28;
	m[0][1]=0xFF;
	m[1][0]=0x03;
	m[1][1]=0xd9;

	c[0][0]=0x09;
	c[0][1]=0xAC;
	c[1][0]=0xB7;
	c[1][1]=0xFb;

	key.SetDims(n, n);
	mInv.SetDims(n, n);
	adjM.SetDims(n, n);
```

```
ZZ mD,invmD;
inv(mD,adjM,m,0);

mD = mD % 256;
if(mD < 0)
        mD+=256;

invmD=InvMod(mD,to_ZZ(256));

for(int i=0;i<n;++i)
{
        for(int j=0;j<n;++j)
        {
                mInv[i][j]=(adjM[i][j]*invmD)%256;
        }
}
mul(key,c,mInv);

cout << "key: " << endl;
my_writeM(key);

ZZ D;

inv(D,adjKey,key,0);
D = D % 256;

if( D<0 )
        D+=256;
invmD=InvMod(D,to_ZZ(256));
```

```
        for (int i = 0; i < n; ++i)
        {       for (int j = 0; j < n; ++j)
                {       invKey[i][j] = (key[i][j]*invmD) % 256;
                }
        }

}


int main()
{
        Mat<ZZ> invKey, bajtM, bajtR;
        int n=2;
        invKey.SetDims(n,n);
        bajtM.SetDims(n,1);
        bajtR.SetDims(n,1);

        key_det(invKey,n);

        cout << "Inv mat: " << endl;
        my_writeM(invKey);


        char *inf="crypHill.txt";
        char *outf="decrypt.jpg";

   ifstream in(inf, ifstream::binary);
   if (!in.is_open()) {
      cout << "Error opening in_file\n";
      return 0;
   }
```

```
    ofstream out(outf, ifstream::binary);
    if (!out.is_open()) {
        cout << "Error opening out_file\n";
        return 0;
    }

            int fsize = in.tellg();
    unsigned char byte1, byte2;

    in.read( (char *) &byte1, 1 );
    in.read( (char *) &byte2, 1 );

            for(int j=0;j<fsize;++j){
                    bajtM[0][0] =byte1;
                    bajtM[1][0] =byte2;
            }

            mul(bajtR, invKey, bajtM);
            out.write(bajtR,fsize);

            in.close();
            out.close();

            return 0;
}
```

// Labor 4 - Feladat 3 Ilyes Hunor-Levente

```
#include<stdio.h>
#include<iostream>
using namespace std;
unsigned char S[256]; unsigned int i, j;
```

```
void swap(unsigned char *s, unsigned int i, unsigned int j)
{
    unsigned char temp = s[i];
    s[i] = s[j];
    s[j] = temp;
}

void rc4_init(unsigned char *key, unsigned int key_length)
{
    for (i = 0; i < 256; i++)
        S[i] = i;

    for (i = j = 0; i < 256; i++)
    {
        j = (j + key[i % key_length] + S[i]) & 255;
        swap(S, i, j);
    }
    i = j = 0;
}
unsigned char rc4_output()
{
    i = (i + 1) & 255;
    j = (j + S[i]) & 255;
    swap(S, i, j);

    return S[(S[i] + S[j]) & 255];
}
```

```
int main()
{
   int k, output_length;
   unsigned char key[] = {"0x1A" "0x2B" "0x3C" "0x4D" "0x5E" "0x6F" "0x77"};
   rc4_init(key, 12);

   cout << "100 lepes: " << endl;
   output_length = 100;
   k = 0;
   while (k < output_length)
   {
      printf("%c", rc4_output());
      k++;
   }

   cout << endl << endl;

   cout << "1000 lepes: " << endl;
   output_length = 1000;

   while (k < output_length)
   {
      printf("%c", rc4_output());
      k++;
   }
}
```

// Labor 4 - Feladat 2 Ilyes Hunor-Levente

```
#include <iostream>
#include <fstream>
using namespace std;
int RC4(char *inf1, char *inf2, char *inf3, char *outf);
```

```
int main() {
    RC4("cryptRC4_Massag", "RC4_Massag.jpg", "cryptHB", "deCrypt.docx");
    return 0;
}

int RC4(char *inf1, char *inf2, char *inf3, char *outf)
{
    ifstream in1(inf1, ifstream::binary);
    if (!in1.is_open()) {
        cout << "Error opening in_file\n";
        return 0;
    }

    ifstream in2(inf2, ifstream::binary);
    if (!in2.is_open()) {
        cout << "Error opening in_file\n";
        return 0;
    }

    ifstream in3(inf3, ifstream::binary);
    if (!in3.is_open()) {
        cout << "Error opening in_file\n";
        return 0;
    }

    ofstream out(outf, ifstream::binary);
    if (!out.is_open()) {
        cout << "Error opening out_file\n";
        return 0;
    }
```

```
    in3.seekg(0, ios::end);
    int fsize = in3.tellg();
    in3.seekg(0, ios::beg);
    unsigned char byte1, byte2, byte3, key, bytes;

    for(int i = 0; i < fsize; i++)
    {
        in1.read( (char *) &byte1, 1 );
        in2.read( (char *) &byte2, 1 );
        in3.read( (char *) &byte3, 1 );
        key     = byte1 ^ byte2;
        bytes   = byte3 ^ key;
        out.write((char *) &bytes, 1);
    }
    out.close();
    in1.close();
    in2.close();
    in3.close();

    return 0;
}
```

// Labor 4 Feladat 5 Ilyes Hunor Levente

```
#include <string>
#include <iostream>
#include <fstream>
using namespace std;
typedef unsigned char uint8;
typedef unsigned int uint32;
uint32 lfsr_output32(uint32 lfsr);
void crypt32(string& c, string m, uint32 lfsr);
```

```
int main()
{
    char *inf = "be.txt";
    ifstream in(inf, ifstream::binary);
    if (!in.is_open())
    {
        cout << "Error opening in_file\n";
        return 0;
    }
    in.seekg(0, ios::end);
    int fsize = in.tellg();
    in.seekg(0, ios::beg);

    char *bytes = new char[fsize];
    in.read(bytes, fsize);

    string m = bytes;
    if (m.length() & 1) m += " ";
    string m1 = "";
    string c = "";
    uint32 lfsr = 0x87354021;

    crypt32(c, m, lfsr);
    cout << "crypted text: " << c << endl << endl << endl;
    crypt32(m1, c, lfsr);
    cout << "decrypted text: " << m1 << endl << endl << endl;
    in.close();
}
```

```
uint32 lfsr_output32(uint32 lfsr)
{
   uint32 bit;
   bit     = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ^ (lfsr >> 7) ^ (lfsr >> 11) ^ (lfsr >> 13) ^
(lfsr >> 17)) & 1;
   lfsr    = (lfsr >> 1) | (bit << 31);
   return lfsr;
}

void crypt32(string &c, string m, uint32 lfsr)
{
   uint32 temp;
   for (int i = 0; i < m.length(); i += 4)
   {
      lfsr = lfsr_output32(lfsr);
      temp = ((uint32)m[i] << 24) | ((uint32)m[i+1] << 16) | ((uint32)m[i+2] << 8) | ((uint32)m[i+3]);
      temp = temp ^ lfsr;

      uint8 st1  = temp >> 24;
      c += st1;

      uint32 st2  = 0x00f0000;
      st2 = (temp & st2) >> 16;
      c += st2;

      uint32 st3  = 0x0000f00;
      st3 = (temp & st3) >> 8;
      c += st3;

      uint32 st4  = 0x000000ff;
      st4 = (temp & st4);
      c += st4; } }
```

```
// Labor 5 - Feladat 1 Ilyes Hunor Levente - Szamitastechnika IV
#define CRYPTOPP_DEFAULT_NO_DLL
#include "dll.h"
#include "aes.h"

USING_NAMESPACE(CryptoPP)
USING_NAMESPACE(std)
//-----------------------------------------------------------------------------------------------------
// AES
//-----------------------------------------------------------------------------------------------------
int main()
{
	byte key[ CryptoPP::AES::DEFAULT_KEYLENGTH ], iv[ CryptoPP::AES::BLOCKSIZE ];
	memset( key, 0x00, CryptoPP::AES::DEFAULT_KEYLENGTH );
	memset( iv, 0x00, CryptoPP::AES::BLOCKSIZE );

	AES::Encryption aesEncryption(key, AES::DEFAULT_KEYLENGTH);
	CBC_Mode_ExternalCipher::Encryption cbcEncryption( aesEncryption, iv );
	FileSource("download.jpg", true, new StreamTransformationFilter(cbcEncryption, new
FileSink("crypt")));

   AES::Decryption aesDecryption(key, AES::DEFAULT_KEYLENGTH);
   CBC_Mode_ExternalCipher::Decryption cbcDecryption( aesDecryption, iv );
	FileSource("crypt", true, new StreamTransformationFilter(cbcDecryption, new
FileSink("kepNew.jpg")));
	return 0;
}
```

```
// Labor 5 - Feladat 1 Ilyes Hunor Levente - Szamitastechnika IV
#define CRYPTOPP_DEFAULT_NO_DLL
#include "dll.h"
#include "aes.h"

USING_NAMESPACE(CryptoPP)
USING_NAMESPACE(std)
//-----------------------------------------------------------------------------------------------------
// 3DES / DES_EDE
//-----------------------------------------------------------------------------------------------------
int main()
{
	byte key[ CryptoPP::DES_EDE3::DEFAULT_KEYLENGTH ], iv[ CryptoPP::DES_EDE3::BLOCKSIZE ];
   memset( key, 0x00, CryptoPP::DES_EDE3::DEFAULT_KEYLENGTH );
   memset( iv, 0x00, CryptoPP::DES_EDE3::BLOCKSIZE );

	DES_EDE3::Encryption desEncryption(key, DES_EDE3::DEFAULT_KEYLENGTH);
	CBC_Mode_ExternalCipher::Encryption cbcEncryption( desEncryption, iv );
	FileSource("download.jpg", true, new StreamTransformationFilter(cbcEncryption, new FileSink("crypt")));

   DES_EDE3::Decryption desDecryption(key, DES_EDE3::DEFAULT_KEYLENGTH);
   CBC_Mode_ExternalCipher::Decryption cbcDecryption( desDecryption, iv );
	FileSource("crypt", true, new StreamTransformationFilter(cbcDecryption, new FileSink("kepNew.jpg")));
	return 0;
}
```

```
// Labor 6 - Feladat 1 Ilyes Hunor Levente
#include <NTL/ZZ.h>
#include <random>
#include <time.h>
#include <ctime>

NTL_CLIENT
using namespace std;

void keyGen(ZZ& e, ZZ& d, ZZ& n, long k);
void enc(unsigned char* &cStr, unsigned char* mStr, ZZ e, ZZ n, long len);
void dec(unsigned char* &mkStr, unsigned char* cStr, ZZ d, ZZ n, long len);

int main()
{
	long k, len, l;
	ZZ e, d, n;
	unsigned char* mStr, *cStr, *mkStr;

	cout << "kulcs meret: ";
	cin >> k;
	keyGen(e, d, n, k);

	cout << "e: " << e << endl;
	cout << "d: " << d << endl;
	cout << "n: " << n << endl;

	len = log(n)/log(to_ZZ(256));
	mStr = new unsigned char [len];
	cStr = new unsigned char [len+1];
	mkStr = new unsigned char [len];
```

```
srand(time(0));
cout << "uzenet hossza: ";
cin >> l;
for (int i = 0; i < l; i++)
{
	mStr[i] = rand() % 256;
	cout << hex << (0xff &mStr[i]) << " ";
}
cout << endl;
if (l > len) {
	cout << "Tul hosszu az uzenet!!";
	return 0;
}
cout << endl << endl;
for (int i = 0; i < 5; i++)
{
	enc(cStr, mStr, e, n, len);
	cout << "titkositott: ";

	for (int i = 0; i < len; i++)
	{
		cout << hex<< (0xff &cStr[i]) << " ";
	}
	cout << endl << endl;
	dec(mkStr, cStr, d, n, len);
	cout << "visszefejtett: ";
	for (int i = 0; i < l; i++)
	{
		cout << hex << (0xff &mkStr[i]) << " ";
	}
```

```
                cout << endl << endl;


        }
        return 0;
}
void enc(unsigned char* &cStr, unsigned char* mStr, ZZ e, ZZ n, long len)
{
        ZZ m, c;
        ZZFromBytes(m, mStr, len);
        cout << "m: " << m << endl << endl;
        PowerMod(c, m, e, n); // c = m^e mod n
        cout << "c: " << c << endl << endl;
        BytesFromZZ(cStr, c, len + 1);


}

void dec(unsigned char* &mkStr, unsigned char* cStr, ZZ d, ZZ n, long len)
{
        ZZ mk, c;
        ZZFromBytes(c, cStr, len + 1);
        cout << "c: " << c << endl << endl;
        PowerMod(mk, c, d, n); // mk = c^d mod n
        cout << "mk: " << mk << endl << endl;
        BytesFromZZ(mkStr, mk, len);
}
void keyGen(ZZ& e, ZZ& d, ZZ& n, long k)
{
        ZZ p, q, phi;
        RandomPrime(p, k/2);
        RandomPrime(q, k/2);
```

```
        n = p * q;
        phi = (p-1) * (q-1);
        e = to_ZZ("65537");
        InvMod(d, e, phi);
}
```