HILL CIPHER

I will first list all of the functions that I use in my code along with an explanation what each function does(some of the function for example the printing ones are just written for easier debugging):

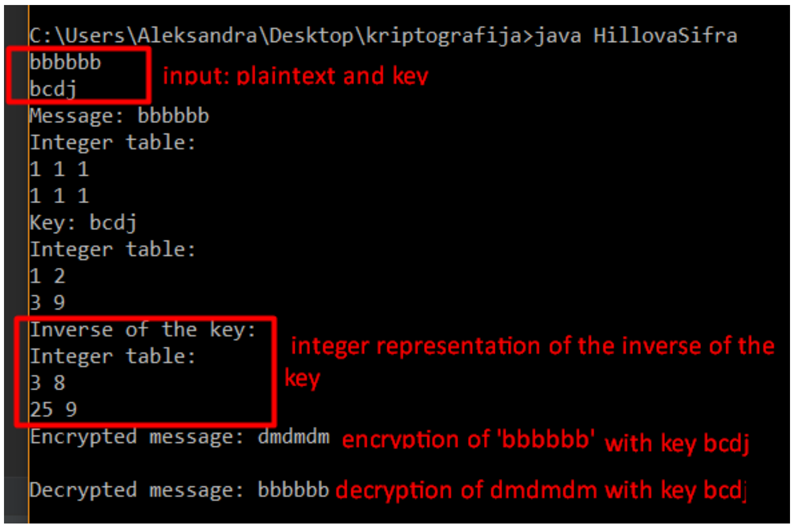
- public static void printInt(int[][] tab) – this function takes a text represented with a matrix and prints it
- public static int[][] CharToInt(String message) -takes a message represented as a string and transforms it into a 2D array with 2 rows and instead of letters it contains a number from 0-25 ('a'=0, 'b'=1, …). It has 2 rows because the key is a matrix of 2x2
- public static int[] CharToInteger(String b) – takes a text and represents it as an integer array, the elements are number from 0-25, each representing the corresponding letter
- public static String IntToChar(int[] b) – takes an array which contains the integer representations of the letters of the message, and transforms it into the corresponding string
- public static void printString(String b) – it prints the string b
- public static void printArray(int[] b) -it prints the message b as integer values
- public static int[][] matrixMultiplication(int[][] m1, int[][] m2) -multiplication of 2 matrices
- public static int inverseNumber(int n) – it returns the multiplicative inverse of n in Z_{26}
- public static int[][] inverseMatrix(int[][] m) – it returns the inverse of a 2x2 matrix. Condition is for the determinant and 26 to have GCD of 1. It is calculated with the formula
- public static double[] frequencies(String message) – it takes a message, and it returns a table of frequencies in the message for each of the letters in the alphabet. For each i from 0 to 25 it calculates how many times it appears in the message, divided by the total number of letters in the message (in %)
- public static double calculate(double[] f1, double[] f2) - returns the sum of the squared differences between the frequencies of each letter in 2 different messages. This is used for calculating "indeks sovpadanja", where we need the sum of the differences between the frequencies of each letter in the ciphertext and in the alphabet (when breaking the Hill cipher)
- public static String Encrypt(int[][] plaintext, int[][] key) – the encryption function. It takes a plaintext and a key which are already represented with matrices and calculates the ciphertext as a matrix (multiplying the message with the key form the left). Then it returns the cipher as a string and not as a matrix. In order to get the string, I had to first to transform the 2D matrix of the cipher into a 1D matrix: I was adding the elements column by column from top to bottom (top right is [0][0]).

```java
public static String Encrypt(int[][] plaintext, int[][] key){
    int[][] rez=matrixMultiplication(key,plaintext);
    char[] kriptogram=new char[rez.length*rez[0].length];
    int k=0;
    for(int i=0; i<rez[0].length; i++)
        for(int j=0; j<2; j++){
            kriptogram[k]=(char)(rez[j][i]+'a');
            k++;
            if(k>=kriptogram.length)
                break;
        }
    return new String(kriptogram);
}
```

- public static String Decrypt(int[][] cryp, int[][]key) – the decryption function. It takes a ciphertext and a key and returns a string of the decrypted message that we got. We get that message with multiplying the cipher with the inverse of the key from the left. The procedure for getting the String is the same as in the encryption function.

```java
public static String Decrypt(int[][] cryp, int[][]key){
    key=inverseMatrix(key);
    int[][] rez=matrixMultiplication(key,cryp);
    char[] kriptogram=new char[rez.length*rez[0].length];
    int k=0;
    for(int i=0; i<rez[0].length; i++)
        for(int j=0; j<2; j++){
            kriptogram[k]=(char)(rez[j][i]+'a');
            k++;
            if(k>=kriptogram.length)
                break;
        }
    return new String(kriptogram);
}
```

Example:

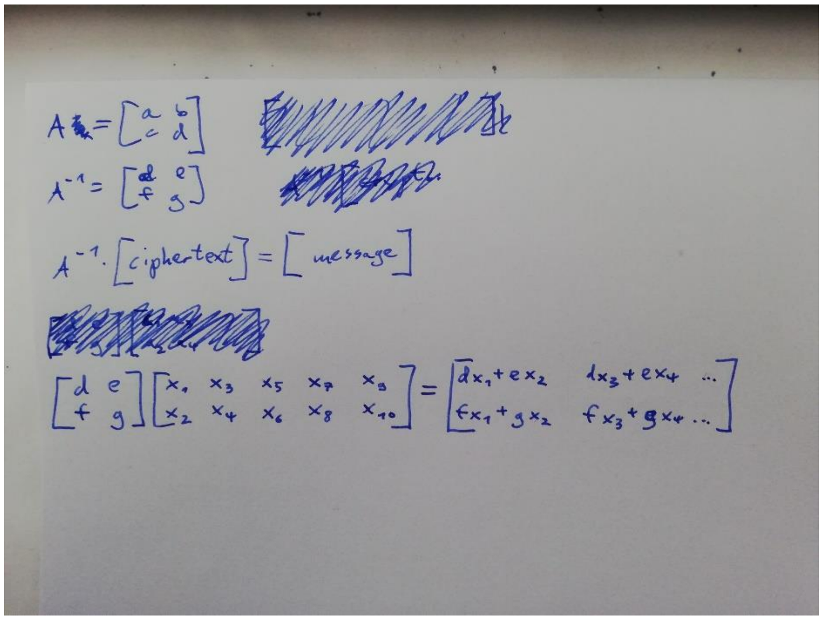


```
C:\Users\Aleksandra\Desktop\kriptografija>java HillovaSifra
bbbbbb
bcdj
Message: bbbbbb
Integer table:
1 1 1
1 1 1
Key: bcdj
Integer table:
1 2
3 9
Inverse of the key:
Integer table:
3 8
25 9
Encrypted message: dmdmdm

Decrypted message: bbbbbb
```

input: plaintext and key
integer representation of the inverse of the key
encryption of 'bbbbbb' with key bcdj
decryption of dmdmdm with key bcdj

- public static String breakHillCipher(int[][] cipher) – Given only the cipher, find the plaintext. What my idea was:
  ciphertext=x1x2x3x4......xl, each xi is an integer representation of a letter


$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
$A^{-1} = \begin{bmatrix} d & e \\ f & g \end{bmatrix}$
$A^{-1} \cdot [\text{ciphertext}] = [\text{message}]$
$\begin{bmatrix} d & e \\ f & g \end{bmatrix} \begin{bmatrix} x_1 & x_3 & x_5 & x_7 & x_9 \\ x_2 & x_4 & x_6 & x_8 & x_{10} \end{bmatrix} = \begin{bmatrix} dx_1 + ex_2 & dx_3 + ex_4 & \dots \\ fx_1 + gx_2 & fx_3 + gx_4 & \dots \end{bmatrix}$

I took that the key is a 2x2 matrix as in the first point. That is the matrix A. For decrypting we need the inverse matrix of the key to get the plaintext. <u>All of the messages that we encrypt or decrypt need to have only 2 rows for them to be able to be multiplied with the key.</u> As we can see in the result in the first row we have only d and e, which is the first row of the key and in the second we have f and e, the second row of the key. So we can divide this problem into two subproblems of the same type, we search for the suitable pairs (d,e) and (f,g). For each pair we have 26x26 possibilities and we try all of them. For each possibility we get one row that represents the a row of the message that we want. We take that row as one substring and calculate the differences between the frequencies between letters in the message and the english alphabet for each letter. We save each result in a 2D array of size [26][26]. Then we take the indexes of the minimal sum to be the values of the pair that we are searching for. When we find the key a.k.a. both pairs, we can decrypt the message with our normal decryption function.

Comments:

-We call the decryption function with the cipher and the inverse of the matrix that we found, because in Decrypt we again do inverse of the key (inverse of inverse is the original key)

-alphabet is an array that contains the frequencies of the letters in the english language (in %)

-Typically, we need a really long english text for the frequencies of the cipher to be close to the ones in the english language

-My program is written to work with lowercase letters

-If the length of the plaintext is not divisable by 2, the letter ‘X’ is added on the end of the string (this is done in main)

-The program can be easily transformed to work for matrices of bigger dimensions-instead of using the formula for finding the inverse of a 2x2 matrix, we will use the Gaussian method.

Additionally, when representing the messages as matrices, we will make them have as many rows as the key has columns.

```
public static String breakHillCipher(int[][] cipher){
    int[][] key1=new int[1][2]; //d,e - first row of inverse matrix of key
    int[][] key2=new int[1][2]; //f,g - second row of index matrix of key
    double[]
alphabet={8.167,1.492,2.202,4.253,12.702,2.228,2.015,6.094,6.966,0.153,1.292,4.025,2.
406,6.749,7.507,1.929,0.095,5.987,6.327,9.356,2.758,0.978,2.560,0.150,1.994,0.077};
    //-------------------------------- d,e:
    double[][] sum1=new double[26][26];
    int[][] rez=new int[cipher.length][cipher[0].length];
    for(int i=0; i<26; i++){
        for(int j=0; j<26; j++){
            key1[0][0]=i;
            key1[0][1]=j;
            rez=matrixMultiplication(key1,cipher);
            double[] freq=frequencies(IntToChar(rez[0]));
            sum1[i][j]=calculate(freq,alphabet);
            if(i==0&&j==0 || i==3&&j==8)
                System.out.println(sum1[i][j]);
        }
    }
    int d=0; //indexes in the 2d array sum1, which represent the values of the first
row of the inverse matrix of the key
    int e=0;
    double minsum=sum1[0][0];
    for(int i=0; i<26; i++)
        for(int j=0; j<26; j++)
            if(minsum>sum1[i][j]){
                minsum=sum1[i][j];
                d=i;
                e=j;
            }
    //-------------------------------- f,g:
    double[][] sum2=new double[26][26];
    int[][] rez2=new int[cipher.length][cipher[0].length];
    for(int i=0; i<26; i++){
        for(int j=0; j<26; j++){
            key2[0][0]=i;
            key2[0][1]=j;
            rez2=matrixMultiplication(key2,cipher);
            double[] freq2=frequencies(IntToChar(rez2[0]));
            sum2[i][j]=calculate(freq2,alphabet);
        }
    }
    int f=0; //indexes in the 2d array sum2, which represent the values of the second
row of the inverse matrix of the key
    int g=0;
    double minsum2=sum2[0][0];
    for(int i=0; i<26; i++)
        for(int j=0; j<26; j++)
            if(minsum2>sum2[i][j]){
                minsum2=sum2[i][j];
                f=i;
                g=j;
            }
    int[][] key=new int[2][2];
    System.out.println();
    System.out.println("d: "+d+", e: "+e+", f: "+f+", g: "+g);
    key[0][0] = d;
    key[0][1] = e;
    key[1][0] = f;
    key[1][1] = g;

    key=inverseMatrix(key);
    return Decrypt(cipher,key);
}
```