CST8234 – C Programming
Assignment 03C: Encryption Algorithms

Hill CipherTransposition Algorithm

A disadvantage of using a simple letter-for-letter substitution cipher is that they preserve the frequency of individual letter, making relatively easy to break the code using a frequency distribution algorithm. One way to overcome this problem is to divide the *plaintex* into group of letters and encipher the *plaintext* group by group. A system of cryptography in which the *plaintex* is divided into groups of *n* letters and then each group is transformed, into a different group of *n* letters, is called **polygraphic** substitution.

The Hill cipher is a **polygraphic** substitution cipher based in matrix transformations. Each character is treated as an integer number from 0 to 255, using the same sequence that the ASCII table, for example:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122

In a simple Hill cipher, group of *n* letters are transformed into the *ciphertext* by the following procedure:

The key to do the encryption is a *n*x*n* matrix with integer entries, for example, a 2 x 2 matrix A is used as the key to do the encryption of the following text:

plaintex: **`olympics`**

$A=\begin{bmatrix}0 & 1\\ 3 & 1\end{bmatrix}$, certain conditions on A has to be imposed as we will discuss later.

The *plaintext* is divided in group of *n*-letters (in this case *n* = 2), and replaced by its numerical value:

o	l	-	y	m	-	p	i	-	c	s
111	108	–	121	109	–	112	105	–	99	115

To encipher the pair ***ol***, we create the matrix product:

$$\begin{bmatrix}0 & 1\\ 3 & 1\end{bmatrix}\begin{bmatrix}111\\ 108\end{bmatrix} = \begin{bmatrix}0*111+1*108\\ 3*111+1*108\end{bmatrix} = \begin{bmatrix}108\\ 441\end{bmatrix}$$

However, there is a problem here since the number 441 has no character equivalent, since our ASCII table has just 256 numbers. To solve this problem, we

make the following agreement:

> *whenever an integer greater that our alphabet – 255 – occurs, it will be replaced by the remainder that results when this integer is divided by 255.*

Thus, we can replaced 441 with 185, which is the remainder after dividing 441 by 256.

As we mention before, the *matrix key* needs to meet certain conditions:

- matrix key needs to have an *inverse* matrix – that will be used to do the decryption.
- Even more, the inverse has to be a *module* m – where m is the size of the alphabet, 256 in our case

You are to write a small C / C++ program to encipher / decipher files – it could be anything, a pdf, an image, a sound file, etc.

Your program is to accept the option -m [matrix_size], the matrix, the file to encrypted and the file to decrypt from the command line argument. Notice that encrypting /decrypting is the same process, what changes is that instead of passing the key, you pass the inverse of the key to do the decryption.

Mystery03, has been encrypted using the following matrix key:

$$A=\begin{bmatrix}0 & 1\\ 3 & 1\end{bmatrix}$$

Using the command:

```
./hill -m 2 0 1 3 1 original Mystery03
```

To properly decrypted, the inverse of the matrix key is:

$$A^{-1}=\begin{bmatrix}85 & 171\\ 1 & 0\end{bmatrix}$$

```
./hill -m 85 171 1 0 Mystery03 file
```