Programming Assignment 1
Hill Cipher

Michael McAlpin

February 4, 2025

1 Hill Cipher

In this assignment you'll write a program that encrypts the alphabetic letters in a file using the Hill cipher where the Hill matrix can be any size from $2x2$ up to $9x9$. Your program will take two command line parameters containing the names of the file storing the encryption key and the file to be encrypted. The program must generate output to the console (terminal) screen as specified below.

1.1 Command line parameters

1. Your program **must compile** and **run** from the **terminal command line**.
2. Input the required file names as command line parameters. Your program may NOT prompt the user to enter the file names. The first parameter must be the name of the encryption key file, as described below. The second parameter must be the name of the file to be encrypted, also described below. The sample run command near the end of this document contains an example of how the parameters should be entered and processed.
3. Your program should open the two files, echo the processed input to the screen, make the necessary calculations, and then output the ciphertext to the console (terminal) screen in the format described below.
4. Your program **must** be named pa01 for whichever language you choose. For example, **pa01**.c, **pa01**.cpp, **pa01**.java, **pa01**.py, **pa01**.rs, or **pa01**.go.

Note
If the plaintext file to be encrypted doesn't have the proper number of alphabetic characters to *match* the key size, pad the last block as necessary with the lowercase letter **x**. Make sure that all the input characters are lower case only.

1.2 Formats

1.2.1 Encryption Key File Formats

The encryption key file will contain a single positive integer, n where $(1 < n < 10)$, on the first line, indicating the number of rows and columns in the encryption matrix. The following n lines will contain n integers, in each row, in order, of the encryption matrix, separated by spaces.

1.2.2 Encryption Plaintext File Formats

The file to be encrypted can be any valid text file with no more than 9,991 letters in it. (Thus, it's safe to store all characters in the file in a character array of size 10,000, including any padding characters.) Please note that the input text file will also generally have punctuation, numbers, special characters, and whitespace in it, which should be ignored. You should also convert uppercase letters to lowercase in the input file, correspondingly lowercase letters do not need to be converted. Thus, the program will treat **A** and **a** the same in your program. Remember that the input plaintext file may need to be *padded to match the block size* of the key.

1.2.3 Output Format

The program must output the following to the console (terminal) screen, also known as `stdout`:

1. Echo the numbers from the input key file. (Hint: Use %4d for the integer formats in the key to align the matrix like the expected outputs in the *base* files.)
2. Echo the lowercase alphabetic text derived from the input plaintext file.
    - Remember to pad with **x** if the processed plaintext does not match the block size of the key.
3. Ciphertext output produced from encrypting the input key file against the input array specified in the key file.

The output portion of the input plaintext file should consist of only lowercase letters in rows of exactly 80 letters per row, except for the last row, which may possibly have fewer. These characters should correspond to the ciphertext produced by encrypting using the numbers collected from the input key file and applied as a Hill cipher, via matrix multiplication (see additional notes and pseudocode at the end of the document). Please note that only the alphabetic letters in the input plaintext file will be encrypted. All other characters should be ignored.

1.2.4 Program Execution

The program, **pa01**, expects two inputs at the command line.

- The first parameter is the name of the key file,which contains a single positive integer, n where ($1 < n < 10$), on the first line, indicating the number of rows and columns in the encryption matrix. The following n lines will contain the contents of each row, in order, of the encryption matrix, separated by spaces.
- The second parameter is the name of the plaintext file, note that this input files may contain non-alphabetic characters (numbers, punctuation, white spaces, etc.). The valid inputs, as discussed previously, may be any alphabetic character, and should be converted to lower case.

1.2.5 Program execution - example

Note that the commands below are also outlined later on in this document for either **c**, **c++**, **Java**, **Go**, or **Python**. Also note that the first parameter is the *key* filename and the second parameter is the *plaintext* filename.

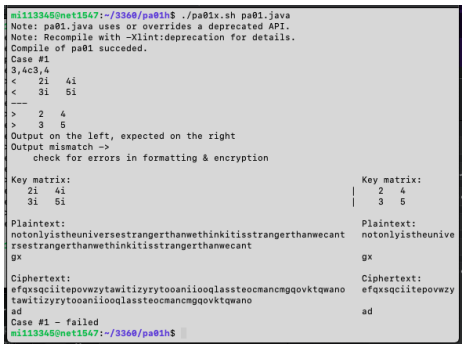
```
systemPrompt$ gcc -o pa01 pa01.c
systemPrompt$ ./pa01 kX.txt pX.txt
```

A sample program execution with outputs is shown below. It is explained in more detail later on in the Section *Sample inputs and outputs* on page 6.

```
systemPrompt$./pa01 k1.txt p1.txt

Key matrix:
   2   4
   3   5

Plaintext:
notonlyistheuniversestrangerthanwethinkitisstrangerthanwecanthinkwernerheisenber
gx

Ciphertext:
efqxsqciitepovwzytawitizyrytooaniiooqlassteocmancmgqovktqwanooqlekytqhkioaawesyt
ad
```

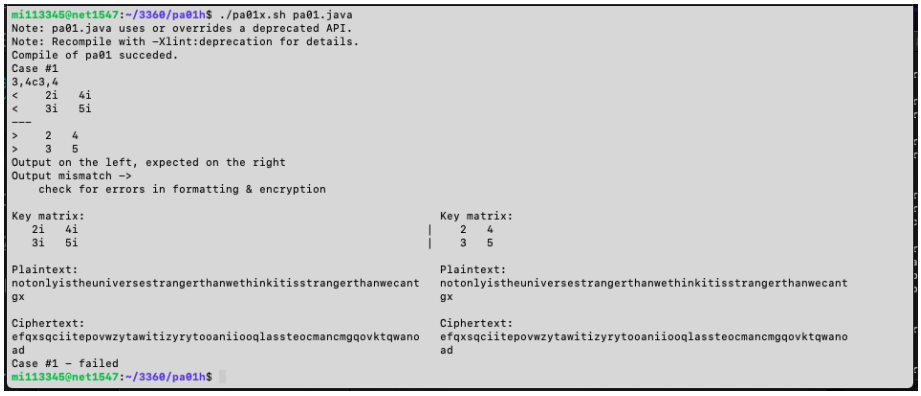
1.2.6 Eustis examples

(a) pa01x.sh with no errors



(b) pa01x.sh with an error in Case #1

Figure 1.1: With a normal putty/ssh width



(a) pa01x.sh with an error in Case #1 - stretched terminal window

Figure 1.2: With a wider or stretched width

1.3 Submission instructions

You must submit this assignment in **Webcourses** as a source file upload. Note that all submissions will be via Webcourses. The submitted programs will be **tested** and **graded** on **Eustis**.

1.3.1 Code Requirements

- Program name - the program name ***must be*** **pa01** in order to work with the shell scripts the graders use to grade your assignment. As mentioned before, the program **must be** named **pa01** for whichever language you choose. For example, **pa01**.c, **pa01**.cpp, **pa01**.java, **pa01**.py, or **pa01**.go.
- Header - the following *Header Usage instructions/comments* comment block should be at the beginning of the source file.

```
/*============================================================================
 |   Assignment:  pa01 - Encrypting a plaintext file using the Hill cipher
 |
 |       Author:  Your name here
 |     Language:  c, c++, Java, go, python, rust
 |   To Compile:  javac pa01.java
 |                gcc -o pa01 pa01.c
 |                g++ -o pa01 pa01.cpp
 |                go build pa01.go
 |                rustc pa01.rs
 |   To Execute:  java   -> java pa01 kX.txt pX.txt
 |           or   c++    -> ./pa01 kX.txt pX.txt
 |           or   c      -> ./pa01 kX.txt pX.txt
 |           or   go     -> ./pa01 kX.txt pX.txt
 |           or   rust   -> ./pa01 kX.txt pX.txt
 |           or   python -> python3 pa01.py kX.txt pX.txt
 |                          where kX.txt is the keytext file
 |                          and pX.txt is plaintext file
 |         Note:
 |                All input files are simple 8 bit ASCII input
 |                All execute commands above have been tested on Eustis
 |
 |        Class:  CIS3360 - Security in Computing - Spring 2025
 |   Instructor:  McAlpin
 |     Due Date:  per assignment
 +============================================================================*/
```

- The following *Academic Integrity Statement* comment block should be at the end of the source file.

```
/*============================================================================
 |     I [your name] ([your NID]) affirm that this program is
 | entirely my own work and that I have neither developed my code together with
 | any another person, nor copied any code from any other person, nor permitted
 | my code to be copied  or otherwise used by any other person, nor have I
 | copied, modified, or otherwise used programs created by others. I acknowledge
 | that any violation of the above terms will be treated as academic dishonesty.
 +============================================================================*/
```

1.4 Program Notes and Hints

Your program must read in an input plaintext file that may contain uppercase letters, lowercase letters and non-letter characters. Your program must distinguish between these three groups so that only the letters get encrypted. All non-letter characters in the file are simply skipped and not counted as part of the plaintext. Please note that although both upper case and lower case letters will be encrypted, your program should convert an upper case input letter to the corresponding lower case letter, i.e., it should convert an **A** to an **a**, and so on for the whole alphabet.
One possible breakdown to solve this problem is as follows:

1. Write a section of code or function that reads only the upper and lower case letters in the input file into an char array of size 10,000, storing only the appropriate lowercase letters in the character array.

2. Write a section of code or function that takes as input the array from section 1 and the encryption key and produces an array of ciphertext storing only lowercase letters.

3. Write a section of code or function that takes as input the array storing the ciphertext and outputs it to the screen in the format specified. Additional functions or code will be needed to echo the input key and plaintext files.

1.5 Sample inputs and outputs

1.5.1 Sample Key File

```
2
2 4
3 5
```

This is k1.txt in the Programming Assignment 1 ZIP file.

1.5.2 Sample Plaintext File

```
 "Not only is the Universe stranger than we think, it is stranger than we
can think."   Werner Heisenberg
```

This is p1.txt in the Programming Assignment 1 ZIP file.

1.5.3 Sample Ciphertext Output File

```
efqxsqciitepovwzytawitizyrytooaniiooqlassteocmancmgqovktqwanooqlekytqhkioaawesyt
ad
```

1.5.4 Sample console output (stdout)

```
systemPrompt$./pa01 k1.txt p1.txt

Key matrix:
   2   4
   3   5

Plaintext:
notonlyistheuniversestrangerthanwethinkitisstrangerthanwecanthinkwernerheisenber
gx

Ciphertext:
efqxsqciitepovwzytawitizyrytooaniiooqlassteocmancmgqovktqwanooqlekytqhkioaawesyt
ad
```

Notes

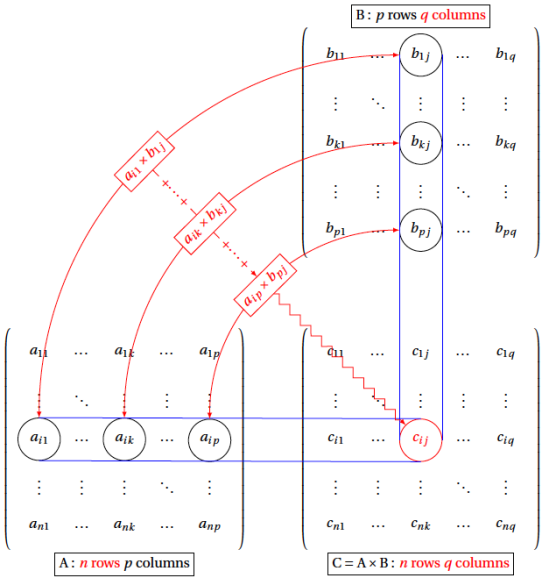
- The file type titles (*Key matrix*, *Plaintext*, & *Ciphertext*) are preceded by a **newline** or `\n`. Likewise after the titles, there is a **newline** or `\n`.
- Each 80 character plaintext or ciphertext line is likewise terminated by a **newline** or `\n`.

2 Matrix multiplication review

A quick review of matrix multiplication:
In mathematics, particularly in linear algebra, matrix multiplication is a binary operation that produces a matrix from two matrices. For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The resulting matrix, known as the matrix product, has the number of rows of the first and the number of columns of the second matrix. The product of matrices **A** and **B** is denoted as **AB**. In the pseudocode below, the product **AB** is **C**.[1]
Remember that in the **Hill Cipher** the key matrix has the same number of columns as rows, and is also known a square matrix.

Algorithm 1 MATRIX-MULTIPLY(A, B)

```
1:  if A.columns ≠ B.rows then
2:      error incompatible dimensions
3:  else
4:      let C be a new A.rows X B.columns matrix
5:      for i = 1 → A.rows do
6:          for j = 1 → B.columns do
7:              c_ij = 0
8:              for k = 1 → A.columns do
9:                  c_ij = c_ij + a_ik × b_kj
10:     RETURN C
```



[1] See also Matrix Multiplication

2.1 Grading

Scoring will be based on the following rubric:

Table 2.1: Grading Rubric

Deduction	Description
-100	Cannot compile on *eustis*
-100	Your program does not successfully compile from the command line with one of these commands: C program: `systemPrompt$gcc -o pa01 pa01.c` C++ program: `systemPrompt$g++ -o pa01 pa01.cpp` Go program: `systemPrompt$go build pa01.go` Java program: `systemPrompt$javac pa01.java` Rust program: `systemPrompt$rustc pa01.rs` Python program: `systemPrompt$python3 pa01.py` Note: *If you are submitting a Java program, the class file must be named "pa01.java" and the class name must be "pa01".* *If you are submitting a Python program, the version of Python running on Eustis is Python 3.*
-100	Cannot read input files specified on command line
-100	Program source filename is **not** **pa01** with the appropriate language extension of **.c**, **.cpp**, **.java**, **.go**, **.py** or **rs**.
-100	Cannot write output to `stdout`
- 90	Your program does not run from the command line without error or produces no output.
- 70	The program compiles, runs, and outputs the key and input file, but crashes thereafter or produces no encryption output.
- 50	Runs without crashing but produces **no meaningful** encryption output
- 20	Fails to produce valid square matrix **key** based on the input **key** file
- 20	Fails to produce valid all lowercase alphabetic **plain text** - formatted to 80 columns per line
- 25	Does not have *Header Usage instructions/comments* statement
- 25	Does not have *Academic Integrity* statement
Start with 100 points and deduct per the schedule above	