HW #1 (Hill Cipher + Playfair)

CIS 3360: Security in Computing

Spring 2026

Instructor: Jie Lin, Ph.D.

Due Date: [**February 16th 2026 by 11:59 p.m.**]

Last updated: February 4, 2026

Disclaimer: This document may not cover all possible scenarios—when in doubt, ask the instructor or a TA.

All official updates (test cases, clarifications, etc.) will be posted as **Webcourses announcements**.

Check Webcourses regularly for critical updates.

Contents

1 Assignment Overview

Implement a program named `hillplayfair` that encrypts a plaintext in two stages:

1. **Stage 1 — Hill cipher:** Encrypt in blocks of size n using an $n \times n$ integer key matrix (with $2 \leq n \leq 9$), arithmetic done modulo 26 with the mapping $A = 0, \ldots, Z = 25$.
2. **Stage 2 — Playfair cipher:** Apply a Playfair digraph cipher (I/J combined) to the result of the Hill stage, using the Playfair keyword provided in the keyword file.

Your program must open the key, plaintext, and keyword files, echo the key matrix and the processed plaintext to the console, perform Hill encryption first, then apply the Playfair cipher, and finally print the ciphertext to the console (format specified below).
All processing is **case-insensitive**. Treat letters `a`–`z` and `A`–`Z` as the same characters, and convert to uppercase internally.
We use the following character-to-number mapping:

Number	Character
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	J
10	K
11	L
12	M
13	N
14	O
15	P
16	Q
17	R
18	S
19	T
20	U
21	V
22	W
23	X
24	Y
25	Z

Retain only alphabetic characters (A–Z), discard all other characters, and if needed pad the final Hill block with the character `X`.

2 Command Line Parameters

Invoke the program from the terminal on the university grading server **Eustis**. The program must accept **exactly four** parameters in this order:

1. The literal command `encrypt`.

2. Path to the **key file**.
3. Path to the **plaintext file**.
4. Path to the **keyword file** (Playfair keyword).

Do not prompt for input; reject incorrect argument counts. Echo the key matrix and the processed plaintext before printing the final ciphertext (see the File Formats section below).

3 File Formats

The assignment specifies strict formats for the input and output.
Quick links:

- Hill cipher key file
- Plaintext file
- Playfair keyword file
- Output format

3.1 Hill Cipher Key File

The key file contains a single integer n (with $2 \leq n \leq 9$) on the first line, indicating the dimension of the square matrix. The next n lines each contain n integers separated by whitespace (spaces or tabs), giving the row-major entries of the Hill key matrix. All arithmetic is done modulo 26.
The key file contains only the $n \times n$ Hill key matrix.
See Sample Key File for an example.

3.2 Playfair Keyword File

The keyword file contains the **Playfair keyword** (alphabetic only). Treat I/J as the same letter when building the Playfair 5×5 square.
See Sample Playfair Keyword File for an example.

3.3 Plaintext File

The plaintext file may contain any characters. Retain only alphabetic letters (A–Z), convert to uppercase, and discard all other characters. If the resulting length is not a multiple of n, append uppercase `X` characters until it is.
See Sample Plaintext File for an example.

3.4 Output Format

Output destination: Print to **standard output** (the terminal) exactly as specified; do not write your results to an output file.
Required sections (in order):

- **Mode:** must display `Encryption Mode`.
- **Original Plaintext:** the raw plaintext file contents exactly as read (no preprocessing, no 80-character wrapping requirement).
- **Hill Cipher Key Dimension:** print the integer n.
- **Hill Cipher Key Matrix:** echo the parsed $n \times n$ matrix (use aligned spacing).
- **Preprocessed Plaintext:** uppercase A–Z only, after preprocessing, wrapped to 80 characters per line.
- **Padded Hill Cipher Plaintext:** the Hill input after padding, wrapped to 80 characters per line.
- **Ciphertext after Hill Cipher:** Hill output, wrapped to 80 characters per line.
- **Playfair Keyword:** the keyword used to build the Playfair table.
- **Playfair Table:** the 5×5 table derived from the keyword (I/J combined).
- **Ciphertext after Playfair:** final ciphertext, wrapped to 80 characters per line.

Line wrapping rule: Only *processed* plaintext/ciphertext sections are wrapped to 80 characters per line; the final line of a section may be shorter.
See Sample Console Output below for the exact format.

4 Related Material Review

4.1 Matrix Multiplication Review

Matrix Multiplication (non-breaking)

Matrix multiplication plays a central role in the Hill cipher. For a block of size n, multiply the $n \times n$ key matrix by the $n \times 1$ column vector of numerical values (with $A = 0$, $B = 1$, …, $Z = 25$) and take all results modulo 26.
For completeness, we include a visual summary of multiplying a matrix A by a vector x.

```
Algorithm 1 MATRIX-MULTIPLY(A,B)
if A.columns != B.rows then
    error "incompatible dimensions"
else
    let C be a new A.rows x B.columns matrix
    for i = 1 -> A.rows do
        for j = 1 -> B.columns do
            c_ij = 0
            for k = 1 -> A.columns do
                c_ij = c_ij + a_ik * b_kj
            end for
            C[i][j] = c_ij
        end for
    end for
    return C
end if
```

Diagram (example): a 4×4 matrix multiplied by a 4×1 vector.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \times \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix}$$

$$\begin{aligned} Y_1 &= A_{11}X_1 + A_{12}X_2 + A_{13}X_3 + A_{14}X_4 \pmod{26} \\ Y_2 &= A_{21}X_1 + A_{22}X_2 + A_{23}X_3 + A_{24}X_4 \pmod{26} \\ Y_3 &= A_{31}X_1 + A_{32}X_2 + A_{33}X_3 + A_{34}X_4 \pmod{26} \\ Y_4 &= A_{41}X_1 + A_{42}X_2 + A_{43}X_3 + A_{44}X_4 \pmod{26} \end{aligned}$$

4.2 Playfair Cipher

Playfair Cipher (example)

Playfair is a digraph substitution cipher using a 5×5 key square built from a keyword (combine I/J). Before encryption, preprocess the text into digraphs as follows:

- Replace `J` with `I`.
- If a digraph would contain two identical letters (e.g., `AA`), insert a padding letter after the first character and then restart pairing from the second character. Use `X` as the padding letter, except when the repeated letter is `X`, in which case use `Z`.
- If one plaintext character is left over at the end, pad the final digraph with `Z`; except if the singleton is `Z` itself, pad with `X`.

Example Playfair key square (keyword: `MONARCHY`):

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Encryption rules for a pair (A,B):

- Same row: replace each with the letter to its right (wrap around).
- Same column: replace each with the letter below it (wrap around).
- Rectangle: replace each with the letter in the same row but the other letter's column.

5 Program Execution

Compile and run on **Eustis**. The only permitted languages are **C, C++, and Rust**. Example commands:

Compilation Commands

```
# C
gcc -O2 -std=c11 -o hillplayfair hillplayfair.c
./hillplayfair encrypt key.txt plain.txt keyword.txt

# C++
g++ -O2 -std=c++17 -o hillplayfair hillplayfair.cpp
./hillplayfair encrypt key.txt plain.txt keyword.txt

# Rust
rustc -O hillplayfair.rs -o hillplayfair
./hillplayfair encrypt key.txt plain.txt keyword.txt
```

6 What We Are Providing

We are providing a zip file called `hillplayfair_test.zip`. In this zip file, you will find the following:

Test File Structure

```
.
+-- expected_results/
|   +-- key01_plaintext01_keyword01.txt
|   +-- key01_plaintext01_keyword02.txt
|   +-- ...
|   +-- key06_plaintext06_keyword06.txt
+-- test_cases/
|   +-- hill_cipher_keys/
|   |   +-- hill_key_01.txt
|   |   +-- ...
|   |   +-- hill_key_06.txt
|   +-- plaintexts/
|   |   +-- plaintext_01.txt
|   |   +-- ...
|   |   +-- plaintext_06.txt
|   +-- playfair_keywords/
|       +-- playfair_keyword_01.txt
|       +-- ...
|       +-- playfair_keyword_06.txt
+-- test_hillplayfair.sh
```

The test cases include all the keys, plaintexts, and Playfair keywords we will use while grading. The folder
`expected_results` contains the expected outputs for the provided test cases.

The
`test_hillplayfair.sh` is an automated shell script that will compile your code and test it against all the expected results. This script supports C, C++, and Rust source files, automatically detects your programming language, compiles your code with appropriate flags, runs all test combinations, and provides detailed feedback on any differences between your output and the expected results.

7 Testing Your Implementation with the Autograder Script (Eustis)

This section explains how to run the provided autograder script locally on **Eustis** before you submit.

Step 0: Put your source file in the correct directory

Place your implementation source file in the **same directory** as `test_hillplayfair.sh`. If the source file is elsewhere, the script may not find/compile it.

Correct Directory Layout (example)

```
Command:
  ls -l

Result (example):
  .
  |-- test_hillplayfair.sh
  |-- expected_results/
  |-- test_cases/
  `-- hillplayfair.c   # OR hillplayfair.cpp OR hillplayfair.rs
```

Step 1: Make the script executable (once)

Setting Script Permissions

```
Command:
  chmod +x test_hillplayfair.sh

Command:
  ls -l test_hillplayfair.sh

Result (example):
  -rwxr-xr-x 1 user group 4321 Feb  5 12:34 test_hillplayfair.sh
```

Step 2: Run the autograder script

Run the autograder script using the same command format for all languages; only the source file name changes.

Running the Autograder (choose ONE command)

```
If you implemented in C, run:
  ./test_hillplayfair.sh hillplayfair.c

If you implemented in C++, run:
  ./test_hillplayfair.sh hillplayfair.cpp

If you implemented in Rust, run:
  ./test_hillplayfair.sh hillplayfair.rs
```

Source file examples (all supported):

- **C:** `hillplayfair.c`
- **C++:** `hillplayfair.cpp`
- **Rust:** `hillplayfair.rs`

The script will compile your code and run the provided test cases, then report a **pass rate** and show any output mismatches.

8 Submission Instructions

Submit on **Webcourses**. Programs are compiled and tested on **Eustis**. Follow these to avoid deductions.

Code Requirements

- **Program name.** Name your program `hillplayfair` with the correct extension for your language (`.c`, `.cpp`, or `.rs`).
- **Header comment (copy/paste).** Place this non-breaking box at the top of your source file (C, C++, or Rust). It will not split across pages and lines will not wrap inside the box.

Required Header Comment

```
/*
Assignment:
hillplayfair - Hill cipher followed by Playfair cipher

Author: <Your Name Here>

Language: C, C++, or Rust (only)

To Compile:
  gcc  -O2 -std=c11   -o hillplayfair hillplayfair.c
  g++  -O2 -std=c++17 -o hillplayfair hillplayfair.cpp
  rustc -O hillplayfair.rs -o hillplayfair

To Execute (on Eustis):
  ./hillplayfair encrypt key.txt plain.txt keyword.txt

where:
  key.txt     = key matrix file
  plain.txt   = plaintext file
  keyword.txt = Playfair keyword file

Notes:
  - Input is text; process A-Z only (case-insensitive).
  - Tested on Eustis.

Class: CIS3360 - Security in Computing - Spring 2026

Instructor: Dr. Jie Lin

Due Date: February 16th 2026
*/
```

What to Submit

- Your source code (can only be .c, .cpp, or .rs file)
- **If you used AI:** Submit the AI Usage Disclosure Form *and* a separate markdown file describing your AI usage.
- **If you did not use AI:** Submit no additional files.

Submission Guidelines

- Submit your source and any required build files on Webcourses before the due date. Late submissions may incur penalties.
- Print to standard output exactly as specified; do not write to additional files.
- Accept exactly four arguments: `encrypt` *key plain keyword*.

9 Sample Inputs and Outputs

Sample Hill Cipher Key File

Sample Key File Content

```
2
2 4
3 5
```

Sample Plaintext File

Sample Plaintext File Content

```
"Not only is the Universe stranger than we think"
```

Sample Playfair Keyword File

Sample Playfair Keyword File Content

```
MONARCHY
```

For complete output format example, see the Sample Console Output section below.

Sample Console Output

The exact strings you print for *Preprocessed Plaintext*, *Padded Hill Cipher Plaintext*, and both ciphertext stages can be very long. For grading, what matters is that your program prints the **same section titles and ordering** and that any **processed** text (preprocessed/padded plaintext and ciphertext outputs) is wrapped at **80 characters per line**.

Sample Program Output (example)

```
systemPrompt$ ./hillplayfair encrypt key.txt plain.txt keyword.txt

Mode:
Encryption Mode

Original Plaintext:
abcdefghijklmnopqrstuvwxyz0123456789
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

Preprocessed Plaintext:
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ

Hill Cipher Key Dimension:
2

Hill Cipher Key Matrix:
   1   0
   0   1

Padded Hill Cipher Plaintext:
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ

Ciphertext after Hill Cipher:
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ

Playfair Keyword:
MONARCHY

Playfair Table:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z

Ciphertext after Playfair:
BIHCFGFYSAKEUCANQSATLZWXWBXRDHCKGIBFKEUCANQSATLZWXWBUZ
```

Note: Every wrapped line can be up to 80 characters; the final line of a section may have fewer. The raw/original file contents are *not* subject to the 80-character wrapping rule.

10 Academic Integrity, AI Usage and Plagiarism Policy

10.1 AI Usage Disclosure

If you use AI tools while completing this assignment, you must disclose this usage. Complete the **AI Usage Disclosure Form** provided with this assignment. If you used AI, include a separate markdown file describing:

- The name and version of the AI tool used.
- The dates used and specific parts of the assignment where the AI assisted.
- The prompts you provided and a summary of the AI output.
- How you verified the AI output against other sources and your own understanding.
- Reflections on what you learned from using the AI.

If you used AI and did not submit both the disclosure form and the markdown file, it will be treated as plagiarism and will result in a 0 for this submission.

10.2 Plagiarism Detection and Writing Standards

All submissions will be processed through plagiarism detection tools (e.g., JPlag). If the similarity score between your submission and others exceeds a threshold, your code will be considered plagiarized and you will receive an F for the course. This applies regardless of whether you used AI and regardless of whether you submitted an AI disclosure form. **If AI usage is detected and the required AI disclosure form/markdown are missing, the submission will be treated as plagiarism and receive a zero.**
While AI tools may assist with brainstorming or high-level guidance, the final submission must represent your own work and understanding. **Do not copy and paste AI-generated code.** AI tends to produce highly similar solutions across students; use it only to understand ideas, then implement the code yourself. Do not copy previous semester solutions or share your code with others.

11 Submission Deadlines and Late Policy

All deadlines use U.S. Eastern Time (Orlando, FL). The Webcourses submission timestamp is authoritative. A submission is considered late if it is uploaded after the posted due date/time. Late submissions are accepted for up to 48 hours after the due date, subject to the following point penalties (deductions are applied to the assignment's maximum score, not a percentage):

- 0:00:01-12:00:00 late → -5 points
- 12:00:01-24:00:00 late → -10 points

- 24:00:01-36:00:00 late → −15 points
- 36:00:01-48:00:00 late → −20 points
- After 48:00:00 → Not accepted; recorded as missed (0 points)

Resubmissions after 48 hours are not accepted.

12 Grading

Your grade for this assignment will be based **primarily** on the **passing rate** of your program against the **provided test cases** and their **expected outputs**, as checked by the provided shell script `test_hillplayfair.sh`.

- If the script reports a **0% pass rate**, then your score for the assignment will be **0**.
- If the script reports an **80% pass rate**, then your score will **most likely be 80 points**, but it is **at most 80 points**.
- If the script reports a **100% pass rate**, you should expect a **perfect score**, barring any additional deductions described below.

Additional deductions may apply. The most common case is **late submission**; see Section 11 for the late policy and the exact point penalties.
Important note about file names on Webcourses: Webcourses may rename files when you resubmit (for example by appending `-1`, `-2`, `-3` before the extension). **This will not result in a deduction.**

Additional Deductions (Non-Pass-Rate Issues)

- **-100 points:** Program cannot compile on Eustis (or does not compile from the command line using the provided compilation commands).
- **-100 points:** Program cannot read the input files specified on the command line.
- **-100 points:** Program name is not `hillplayfair` with the appropriate language extension (`.c`, `.cpp`, or `.rs`).
- **-100 points:** Program does not accept exactly four arguments in the required order (`encrypt` *key plain keyword*) or does not run from the command line.
- **-100 points:** Required header comment block is missing or not properly updated (for example, if `Author:` does not include your name).
- **-100 points:** Plagiarism / academic integrity violations, including (but not limited to) any of the following:

  - Using AI **without** submitting the required AI disclosure form and the required markdown description file.
  - Evidence of direct copying from AI output or another student.
  - Very high similarity to another submission (e.g., flagged by automated similarity checking).

  See Section 10.

Aside from the items above, grading is driven by the shell-script pass rate. Follow all formatting and output requirements precisely.

13 References

- L. S. Hill, “Cryptography in an Algebraic Alphabet,” *The American Mathematical Monthly*, vol. 36, no. 6, 1929.
- “Hill cipher,” Wikipedia, `https://en.wikipedia.org/wiki/Hill_cipher`
- “Playfair cipher,” Wikipedia, `https://en.wikipedia.org/wiki/Playfair_cipher`