PROGRAMMING ASSIGNMENT 4

TAs **:** Nebi YILMAZ, Bahar GEZİCİ

Due Date **: 25.12.2020 (23:00)**

Click here to accept your Programming Assignment 4.

1. Introduction

In this assignment, you will get familiar with IO operations, functions, matrices, testing, and design a relatively complex algorithm. For this purpose, you will write a text encryption / decryption tool that can extract plain / ciphertext and key from files, encrypt / decrypt them and return the results to a file.

2. BackGround

2.1. Encryption

Encryption was first used 4,000 years ago. Historical findings prove that Egyptians, Indians, Chinese, and ancient Greeks encrypted messages. The most well-known historical encryption technique is the relocation technique called the Caesar cipher used by Julius Caesar. The technique involves changing letters in the alphabet to encode a message (such as A->C, B->D). Nowadays, the replacement technique is too easy to be considered safe.

Figure 1 The Enigma Machine

20. in the century, the application of mathematical theory and methods to encryption began to take place in military use. In the Second World War, encryption played an important role. He used special coding machines for encryption at the time. Alan Turing, a pioneering computer scientist in the field of artificial intelligence, was able to decrypt messages encrypted by the German Enigma machine, and was able to use it in World War II. He invented a machine that helped change the course of World War II.

With the advent of the computer age and internet communications, the use of encryption has become widespread in communications and in securing private data; it is no longer limited to military uses. Modern encryption methods are more complex and often combine several steps or methods to encrypt the data, making the data more secure and more difficult to break. Some modern methods use matrices as part of encryption and decryption; other areas of mathematics, such as number theory, play a major role in modern cryptography.

2.1. Assertion

There are many test methods for Python. The easiest method is assertion. You should use different assertions to detect errors in inputs, parameters and outputs. There are some general best practices around how to write assertions:

Method	Equivalent to
.assertEqual(a, b)	a == b
.assertTrue(x)	bool(x) is True
.assertFalse(x)	bool(x) is False
.assertIs(a, b)	a is b
.assertIsNone(x)	x is None
.assertIn(a, b)	a in b
.assertIsInstance(a, b)	isinstance(a, b)

.assertIs(), .assertIsNone(), .assertIn(), and .assertIsInstance() all have opposite methods, named .assertIsNot(), and so forth.

3. Problem

In this assignment, you are going to develop an encryption method that uses matrix multiplication and matrix inversions. Your procedure for using matrices when encoding and decoding hidden messages is as follows.

First convert the secret message into a string of numbers by arbitrarily assigning a number to each letter of the message. Next, convert this string of numbers into a new set of numbers by multiplying the string

by a square matrix of our choice that has an inverse. This new set of numbers represents the coded message.

To decode the message, take the string of coded numbers and multiply it by the inverse of the matrix to get the original string of numbers. Finally, by associating the numbers with their corresponding letters, obtain the original message.

You will use the correspondence shown in following Table where letters A to Z correspond to the numbers 1 to 26 and a space is represented by the number 27.

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

3.1. Example of Encoding

Encode the message: ATTACK NOW by using following key matrix *A*;

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$$

We divide the letters of the message into groups of two. “AT”,“TA”,”CK”,”-N”,”OW”

Here, we divide the letters of the message into groups of two because size of key matrix is 2x2. (For example, if the size of key matrix was 3x3, we should divide the letters of the message into groups of three. In other words, this situation changes according to the matrix size).

We assign the numbers to these letters from the above Table, and convert each pair of numbers into 2x1 matrices (For example, if the size of matrix was 3x3, we should convert each pair of numbers into 3x1 matrices). In the case where a single letter is left over on the end, a space is added to make it into a pair.

$$\begin{bmatrix} \text{A} \\ \text{T} \end{bmatrix} = \begin{bmatrix} 1 \\ 20 \end{bmatrix} \quad \begin{bmatrix} \text{T} \\ \text{A} \end{bmatrix} = \begin{bmatrix} 20 \\ 1 \end{bmatrix} \quad \begin{bmatrix} \text{C} \\ \text{K} \end{bmatrix} = \begin{bmatrix} 3 \\ 11 \end{bmatrix} \quad \begin{bmatrix} \text{–} \\ \text{N} \end{bmatrix} = \begin{bmatrix} 27 \\ 14 \end{bmatrix} \quad \begin{bmatrix} \text{O} \\ \text{W} \end{bmatrix} = \begin{bmatrix} 15 \\ 23 \end{bmatrix}$$

So at this stage, our message expressed as 2x1 matrices is as follows.

$$\begin{bmatrix}1\\20\end{bmatrix}\begin{bmatrix}20\\1\end{bmatrix}\begin{bmatrix}3\\11\end{bmatrix}\begin{bmatrix}27\\14\end{bmatrix}\begin{bmatrix}15\\23\end{bmatrix} \quad \textbf{(1)}$$

Now to encode, we multiply, on the left, each matrix of our message by the matrix A. For example, the product of A with our first matrix is:

$$\begin{bmatrix}1 & 2\\1 & 3\end{bmatrix}\begin{bmatrix}1\\20\end{bmatrix}=\begin{bmatrix}41\\61\end{bmatrix}$$

```
matrix1 = [[1,2],[1,3]]
matrix2 = [[1],[20]]
result  = [[0],[0]]
for i in range(len(matrix1)):
    for j in range(len(matrix2[0])):
        for k in range(len(matrix2)):
            result[i][j] += matrix1[i][k] * matrix2[k][j]
print(result)
```

And the product of A with our second matrix is:

$$\begin{bmatrix}1 & 2\\1 & 3\end{bmatrix}\begin{bmatrix}20\\1\end{bmatrix}=\begin{bmatrix}22\\23\end{bmatrix}$$

Multiplying each matrix in **(1)** by matrix A, in turn, gives the desired coded message:

$$\begin{bmatrix}41\\61\end{bmatrix}\begin{bmatrix}22\\23\end{bmatrix}\begin{bmatrix}25\\36\end{bmatrix}\begin{bmatrix}55\\69\end{bmatrix}\begin{bmatrix}61\\84\end{bmatrix}$$

“ATTACK NOW!” ⇒ “41,61,22,23,25,36,55,69,61,84”

Plain text — Encrypted text

3.2. Example of Decoding

Decode the “41,61,22,23,25,36,55,69,61,84” message that was encoded using following matrix A.

$$\begin{bmatrix}41\\61\end{bmatrix}\begin{bmatrix}22\\23\end{bmatrix}\begin{bmatrix}25\\36\end{bmatrix}\begin{bmatrix}55\\69\end{bmatrix}\begin{bmatrix}61\\84\end{bmatrix} \quad \textbf{(2)} \qquad A=\begin{bmatrix}1 & 2\\1 & 3\end{bmatrix}$$

Firstly, we grouped the secret message as a matrix in two because size of key matrix is 2x2 (The number of elements grouped as matrix varies according to the key matrix size as in encoding.)

Since this message was encoded by multiplying by the matrix *A*. First determine inverse of *A*

$$A^{-1} = \begin{bmatrix} 3 & -2 \\ -1 & 1 \end{bmatrix}$$

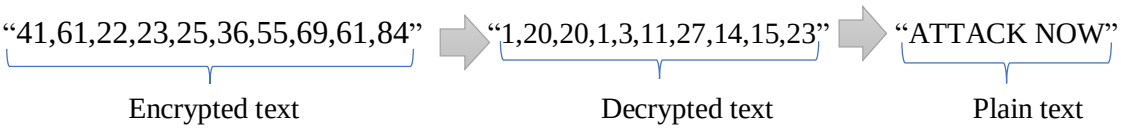
To decode the message, multiply each matrix, on the left, by A^{-1}. For example

$$\begin{bmatrix} 3 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 41 \\ 61 \end{bmatrix} = \begin{bmatrix} 1 \\ 20 \end{bmatrix}$$

Multiplying each of the matrices in **(2)** by the matrix A^{-1} gives the following.

$$\begin{bmatrix} 1 \\ 20 \end{bmatrix} \begin{bmatrix} 20 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 11 \end{bmatrix} \begin{bmatrix} 27 \\ 14 \end{bmatrix} \begin{bmatrix} 15 \\ 23 \end{bmatrix}$$

Finally, by associating the numbers with their corresponding letters, obtain



4. Execution and Test

Your program will be tried many times with different combinations of inputs. In addition to normal entries, incorrect entries will be given because of "Testing" within the scope of this assignment. It is recommended to use Map and Reduce Functions for this assignment.

Your program must be parametric. You are expected to take operation type and files as a parameter as shown below (nonparametric programs will not be evaluated as they cannot be tested):

python3 assingment4.py [operation type: enc / dec] "key file path" "input file path" "output file name"

As a parameter, you will use operation type, file path for *key file* and *input file* (*plain_input.txt* or *ciphertext.txt) and* name of the output file for *output file* that you will create.

You will use input file that named *plain_input.txt* for encoding and *ciphertext.txt* for decoding as shown in Section 5 to test your code.

Note: When you test your code with *plain_input.txt*, you program will create an output file named *output_enc.txt* into same file with you source code.

Note: When you test your code with *ciphertext.txt*, you program will create an output file named *output_dec.txt* into same file with you source code.

Your program will be evaluated in 3 different aspects: encryption, decryption and assertion.

4.1. Encryption

Your program should read the key and input files from which it learns paths from parameters. First of all, the plain text in the input file should be digitized as described above. Afterward, the text should be encrypted by multiplying it with the key matrix in the key file.

Different sizes of key matrices can be used for both encryption and decryption.

4.2. Decryption

Your program should read the key and input files from which it learns paths from parameters. First of all, the ciphertext in the input file should be read. Afterward, the ciphertext should be decrypted by multiplying it with inverse of the key matrix in the key file.

4.3. Assertion

Your program should not crash when incorrect entries or commands are given. It should identify the error and write the error message on the screen. Your program will be tested by selecting some of the errors listed below.

1- **"Parameter number"** error: Occurs when the program has different numbers of parameters than 4.
2- **"Undefined parameter"** error: Occurs if a value other than "enc" or "dec" is entered in the operation type parameter.
3- **"Input file not found"** error: Occurs when the input file cannot be found in the path specified by the parameter.
4- **"The input file could not be read"** error: Occurs when the format of the input file specified by the parameter is corrupt.
5- **"Input file is empty"** error: Occurs when the input file specified by the parameter is empty.
6- **"Invalid character in input file"** error: Occurs when the input file specified by the parameter contains characters other than 26 letters and space.
7- **"Key file not found"** error: Occurs when the key file cannot be found in the path specified by the parameter.

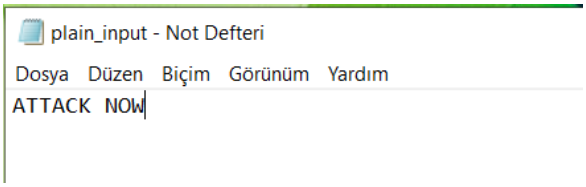
8- **"Key file could not be read"** error: Occurs when the format of the key file specified by the parameter is corrupt.

9- **"Key file is empty"** error: Occurs when the input file specified by the parameter is empty.

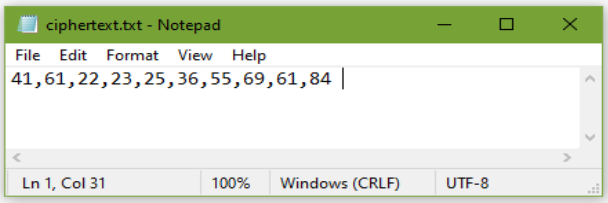
10- **"Invalid character in key file"** error: Occurs when there are characters other than numbers and comma in the key file specified by the parameter.

5. Input and Output File Format

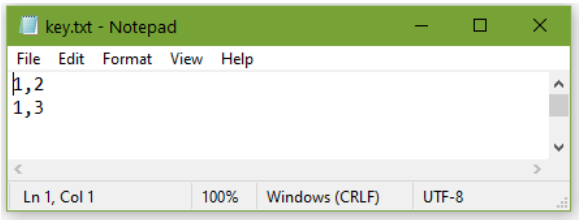
Plain input file: 26 characters from A to Z and spaces (other characters may be included to test the assertion mechanism)



Ciphertext file: Comma-separated positive integers (other characters may be included to test the assertion mechanism)



Key file: It will be always be a square matrix such as 2x2, 3x3, ..., nxn matrix. Comma-separated positive integers and \n (newline) character. (other characters may be included to test the assertion mechanism)



6. Grading Policy

Task	**Point**
Coding style and standard	5
Encryption	35
Decryption	35
Assertion	25
Total	**100**

7. Important Notes

- Do not miss the submission deadline.
- Compile your code on dev.cs.hacettepe.edu.tr before submitting your work to make sure it compiles without any problems on our server.
- You do not allow to use Numpy library.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating. You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza. You cannot share algorithms or source code. All work must be individual! Assignments will be checked for similarity, and there will be serious consequences if plagiarism is detected.
- You may assume that the input file will be given as command-line arguments, so to execute your code on dev use the following command in your terminal:

  For encoding run your code as follow:

  python3 assignment4.py enc ../folder_x/key1.txt ../folder_y/plain_input.txt output_enc.txt

  For decoding run your code as follow:

  python3 assignment4.py dec ../folder_x/key1.txt ../folder_y/ciphertext.txt output_dec.txt
- You must submit your work with the file as stated below:

  assignment4.py