Lab 04

Instructor: Shahriar Ivan

General instructions:

Create a java application project naming the java file as Lab04_2A_ID where ID is your student ID. The example snippets will use the general class name Lab04_2A without the ID portion. If there are multiple tasks, you don't have to create separate projects for each task. A single project file should contain all the .java files that would be necessary to satisfy all the tasks given here.

Alternately, you can submit just the required .java files (without the project). Be sure to name the .java file with the main function as Lab04_2A_ID.java.

Tasks:

Today's lab task will focus on Java File I/O. Normally we handle most input-output operations using the standard input (keyboard) and the standard output (monitor) stream. But if required, we can switch to a different I/O stream so that the program takes input from a different source (such as a file) rather than the keyboard and also writes output to a different destination (such as another file) rather than the monitor. For this, Java has a convenient package which we import here as java.io.*. Here the '*' character at the end signifies that we are importing "all" the classes under the io package.

The following is the public class that contains the main function:

```
import java.io.*;
import java.util.Scanner;


public class Lab04 {
    public static String decrypt_vigenere(String c, String k){
          // need to implement this function
    }


    public static void main(String[] args) {
        try{
            File in1 = new File
("C:\\Users\\Acer\\Documents\\NetBeansProjects\\Lab04\\src\\lab04\\input.txt");
   // you may have to change this part depending on your file location
            Scanner fileReader1 = new Scanner(in1);
            String ciphertext = "";


            while(fileReader1.hasNextLine()){
```

```
                ciphertext = ciphertext + fileReader1.nextLine() + "\n";
            }

            System.out.println("Ciphertext:\n" + ciphertext);

            fileReader1.close();

            File in2 = new File
("C:\\Users\\Acer\\Documents\\NetBeansProjects\\Lab04\\src\\lab04\\key.txt");
   // you may need to change this part depending on your file location
            Scanner fileReader2 = new Scanner(in2);
            String key = "";

            while(fileReader2.hasNextLine()){
                key = key + fileReader2.nextLine() + "\n";
            }

            System.out.println("Key:\n" + key);

            String plaintext = decrypt_vigenere(ciphertext, key);    //Implement
the decrypt function yourself
            System.out.println("Decrypted plaintext:\n"+plaintext);

            //Logic for writing the plaintext into another file named output.txt

        }
        catch(FileNotFoundException e){
            System.out.println("An error occurred while processing file");
            e.printStackTrace();
        }
    }
}
```

You can copy paste the above code for the class with the main function. For today's task, you will not need to implement any other classes.

There are **two** things you need to implement here. First is the **decrypt_vigenere** function which takes two strings, **ciphertext** and **key** each of which are obtained from separate .txt files given with the assignment. Second thing is to implement the code that writes the decrypted text into another file called "output.txt" that will be created in the same directory

as the input and key file. Look up some examples for the **FileWriter** class and the **write()** method for implementing this part. Your task is to submit the java file with the code as well as the output.txt file that you generated.

> **Note1: The string given for the filename may need to be changed based on your file location. Notice that an extra backslash character(\) is used after every backslash character in the filename. This is used as an escape character since \ has a different meaning in java language.**
>
> **Note2: For simplicity, you can consider the plaintext, key and ciphertext are all in UPPERCASE letters. But if you want, you can handle lowercase characters as well with the same logic. Also anything other than alphabets, such as space, newline and special characters are unencrypted and can be ignored during the decryption process since they will stay as they are.**

In order to implement the **decrypt_vigenere()** function, we need to know how the Vigenere cipher encryption works. Then your task would be to simply reverse this process to get the decryption mechanism.

Method of Vigenere encryption

Vigenere cipher is a classical cipher that uses a series of interwoven Caesar ciphers.

Suppose we have a plaintext, P = ATTACK AT DAWN!

And suppose the key, K = LEMON

Then the encryption process of Vigenere cipher would repeat this key until it matches the length on the plaintext, and then it would simply add the character-wise positions to get the corresponding characters in the cipher text.

Here, A=0, B=1, …, Z=25

So, A (value=0) + L (value=11) = L (value=11)

Similarly, T (value=19) + E(value=4) = X (value=23)

And so on…

Thus we get the whole encryption process as:

P = ATTACK AT DAWN!

K = LEMONL EM ONLE!

C = LXFOPV EF RNHR!

Notice that space and special characters like exclamation mark (!) are not processed and kept as is.

The process we just observed converts a plaintext to a ciphertext. But in this assignment we have the ciphertext and the key, and the algorithm used to obtain the ciphertext has been mentioned. Your task is to reverse this algorithm to get the plaintext.