GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computing & Mathematics

Data Structures & Algorithms

ASSIGNMENT DESCRIPTION & SCHEDULE

Rapid Encryption using the German WW1 ADFGVX Cypher

Note: This assignment will constitute 50% of the total marks for this module.

1. Overview

You are required to develop a Java application that is capable of encrypting and decrypting a text using an ADFGVX cypher. The ADFGVX cypher was used by the German Army in WW1 from March 1918 to encrypt field communications during the Ludendorff Offensive (Kaiserschlacht). The cypher is so named because all messages are encrypted into codes from the small alphabet of ADFGVX to reduce operator error when sending Morse Code signals. Although an improvement on the ADFGX cypher used by the Germans up until 1918, the new cypher was broken by the French cryptanalyst Georges Painvin and proved decisive in repulsing the attack at Compiègne on June 11th 1918.

The ADFGVX cypher uses a Polybius square to encode each letter / number as two symbols in the alphabet {ADFGVX}. This is not unlike a Vigenère / polyalphabetic cipher.

	A	**D**	**F**	**G**	**V**	**X**
A	P	H	0	Q	G	6
D	4	M	E	A	1	Y
F	L	2	N	O	F	D
G	X	K	R	3	C	V
V	S	5	Z	W	7	B
X	J	9	U	T	I	8

Fig 1. Polybius Square

In addition to the Polybius square, the ADFGVX cypher also requires a code word to be used to diffuse and fractionate the codes derived from the square.

2. Encrypting a Message

1. To encode a plaintext character using the Polybius Square, locate the character in the matrix and read off the letter on the ***far left side on the same row*** and then the letter at the ***top of the same column***, i.e. *each plaintext character is represented by two cipher characters*. For example, the plaintext "OBJECT" will generate the sequence of pairs {FG, VX, XA, DF, GV, XG};

2. ***Create a matrix*** from a code word with the enciphered codes underneath. In this case the code word *JAVA* will be used for both encryption and decryption. As each plaintext character has is represented by two enciphered codes, it creates a degree of *diffusion* in the cypher.

J	A	V	A
F	G	V	X
X	A	D	F
G	V	X	G

3. Perform a ***columnar transposition***, by sorting the plaintext alphabetically. This steps *fractionates* the cypher by splitting up the two enciphered codes associated with each plaintext character.

A	A	J	V
G	X	F	V
A	F	X	D
V	G	G	X

4. The final cyphertext is formed by ***reading off each column***:

{GAVXFGFXGVDX}

3. Decrypting a Message

The decryption of a message requires that the columnar transposition in Step 3 above is undone. The can be performed as by reading each set of codes into the correct column denoted by the index and then checking each pair of code along each row against the Polybius Square.

J	A	V	A
2	*0*	*3*	*1*
F	G	V	X
X	A	D	F
G	V	X	G

Note that, because the encrypted message is 12 characters long and the code word JAVA is four characters long, we can compute the number of rows required in the decoding matrix as the *message-length/code-word-length*. Use the modulus operator (%) to test if there is a remainder – if so, an additional row will be required.

4. Minimum Requirements

You are required to develop a Java application that can parse and encrypt / decrypt the contents of a text file (small or very large…) using the ADFGVX.

- The application should prompt the user to enter a key phrase (code word) and a file or URL and then parse the given resource line-by-line, encrypt / decrypt the text and append it to an output file.

- You must comment each method in your application stating its running time (in Big-O notation) and your rationale for its estimation.

Please note that extra marks will be given for the appropriate application of the ideas and principles we study on this course. The emphasis should be on speed (low time complexity) and a minimal amount of RAM consumption (low space complexity)

5. Deployment and Delivery

- ***The project must be submitted by midnight on Sunday 5th April 2015.*** The project must be submitted as a Zip archive ***(not a rar, 7-zip or WinRar file)*** using the Moodle upload utility. You can find the area to upload the project under the heading *"Rapid Encryption using the German WW1 ADFGVX Cypher (50%): Assignment Upload"* in the "Notices and Assignments" section of Moodle.
- The name of the Zip archive should be <*id*>.zip where <*id*> is your GMIT student number.
- The Zip archive should have the following structure (do NOT submit the assignment as an Eclipse project):

Marks	Category
src	A directory that contains the packaged source code for your application. Use the package name ***gmit*** for your classes.
README.txt	A text file outlining the main features of your application, with a set of instructions for running the programme.
encoder.jar	A JAR archive containing a fully functional version of the application. You can create a JAR archive from inside you Eclipse project's bin directory with the following command line instruction: ***jar –cf encoder.jar gmit/*.class***
TestRunner.java	A simple Java class that demonstrates the functionality of the encoder application. This class should include the import: import gmit/*;

Note: extra marks will be awarded for additional ***relevant*** features and functionality. Remember – this is a course on data structures and algorithms!

6. Copying & Plagiarism

Please note that the college policy on plagiarism is set out in the Student Code of Conduct (see http://www.gmit.ie/presidents-office/quality-assurance/academic-codes-practice/code-of-practice-no7.pdf). Plagiarism is the passing off of the work of another person as one's own and constitutes a serious infraction that may result in disciplinary proceedings. This assignment is an INDIVIDUAL assignment. While collaboration with other class members is acceptable for high-level design and general problem solving, you must individually code, implement and document your own submission.