CSCI3180 – Principles of Programming Languages – Spring 2012

Course Project Phase 1 Exercise — Vigenère Cipher

Phase 1 Deadline: Mar 4, 2012 (Sunday) 23:59

1 Introduction

This is a simple programming exercise for Course Project Phase 1 in order to help you get familiar with C, Smalltalk and the corresponding development environment. In this exercise, you have to implement the Vigenère Cipher using C and Smalltalk. Vigenère Cipher is based on a series of Caesar ciphers and utilize a secret key to produce the ciphers.

2 Exercise Details

In cryptography, encryption is the process of transforming input information (**plaintext**) using an algorithm (**cipher**) to make it unreadable to anyone except those possessing special knowledge (**key**). The result of the process is encrypted information (**ciphertext**). The reverse process, i.e. to make the encrypted information readable again is referred to as decryption.

2.1 Caesar Cipher

Caesar cipher is one of the simplest and widely known encryption methods. In Caesar cipher, each alphabetical character in the plaintext is shifted with some number of positions. Encryption is NOT performed on any non-alphabetical characters and the original characters are returned. For example, if we encrypt "JIMMY" by right-rotating it by three positions, we will obtain "MLPPB". As we can see in the example, we right-rotate "Y" by three positions and obtain "B" in the cipher. With a right-rotate of three positions, we encrypt the letters using the following mapping:

plaintext:	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
ciphertext:	`DEFGHIJKLMNOPQRSTUVWXYZABC`

2.2 Vigenère Cipher

Vigenère Cipher is based on a series of Caesar ciphers and use a secret key to get the exact cipher among the series of Caesar ciphers. The series of Caesar ciphers are defined in the Vigenère Table (Table 1), which is a 26x26 matrix and each row is left-rotated one place compared to the upper row.

The encryption is done by mapping the plaintext letter and the key letter to the ciphertext letter by using the Vigenère Table. Each row represents a key letter and each column represents a plaintext letter. These two letters map to another ciphertext letter in the table. For example, with key letter 'K' and plaintext letter 'O', we will obtain the ciphertext letter 'Y'. As in Caesar cipher, all non-alphabetical letters will not be accessed and the original characters will be returned.

The encryption works as follows: you are given a plaintext and a secret key, the ciphertext is generated by substituting all alphabetical letters of the plaintext by their ciphertext letters. All alphabetical letters of the plaintext are regarded as **uppercase** letters. For example, in case 1 of Table 2, "Angry" is treated as "`ANGRY`".

1. The key letter to generate each ciphertext letter is achieved by **mapping** the key to the alphabetical characters of plaintext letter by letter. The ciphertext letter is produced by looking up the Vigenère Table. A sample is shown in case 1 of Table 2.
2. When the length of the key is less than that of the plaintext, the key is **repeated** until the end of plaintext, as in case 2 of Table 2.

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

Table 1: The Vigenère Table

3. Note that **only alphabetical letters** are mapped and substituted. Non-alphabetical characters should be copied directly to the ciphertext correspondingly. As shown in case 3 of Table 2 (space, digits and '`!`' are skipped). Key letters are not mapped to non-alphabetical characters.

Case	plaintext	Key	Plaintext-Key Mapping	ciphertext
1	Angry	JimmyLee	`ANGRY` `JIMMY`	`JVSDW`
2	Programming	Funny	`PROGRAMMING` `FUNNYFUNNYF`	`ULBTPFGZVLL`
3	I like CSCI3180 too!	MeToo	`I LIKE CSCI3180 TOO!` `M ETOO METO OME`	`U PBYS OWVW3180 HAS!`

Table 2: Vigenère Cipher Examples

Decryption is the reverse operation of encryption. In decryption, the ciphertext and the secret key are taken as input and the produced plaintext as output. The one-to-one letters mapping operation is the same as in encryption, except that the plaintext becomes ciphertext. Then, the substitution works as follows:

1. Look up the row correspond to the key letter;
2. Look up the ciphertext letter in that row;
3. Return the corresponding column letter as the plaintext letter.

For example, if we have a ciphertext letter 'C' and key letter 'K', we first look up the row corresponding to letter 'K' in the Vigenère Table and find it is the 11^{th} row. Then we look up the letter 'C' in row 11 and find that it is at the 19^{th} column, which corresponds to letter 'S'. So, the plaintext letter is 'S'.

3 General Specification

You are required to write programs using C and Smalltalk to implement Vigenère Cipher. Your program should be able to: 1) take a plaintext and a secret key as input and produce the cor-

responding ciphertext as output; 2) take a ciphertext and a secret key as input and produce the corresponding plaintext as output;

1. **Smalltalk Classes**
   Please follow the class defined below in the Smalltalk implementation. You can add other variables, methods or classes as you like.

   Class `VigenereCipher`
   The class used to perform encryption and decryption using Vigenère Cipher. You have to implement it with the following components:

   (a) **Instance Variable(s)**

   `private key`
   - This is a string of key used to do encryption or decryption.

   (b) **Instance Method(s)**

   `public key: aKey`
   - Setting the secret key used to do encryption or decryption.

   `public encrypt: aStr`
   - Perform encryption using the key on the plaintext (aStr) and return the string of ciphertext.

   `public decrypt: aStr`
   - Perform encryption using the key on the ciphertext (aStr) and return the string of plaintext.

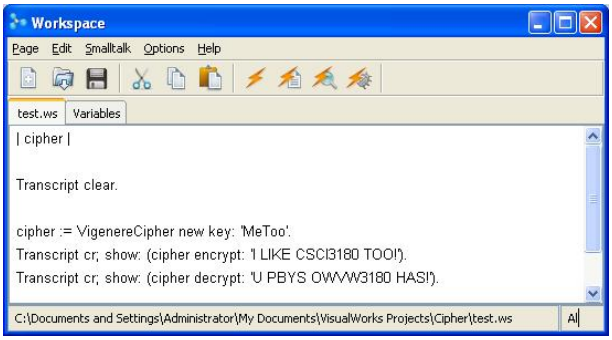
2. **Input Specification**
   In the C implementation, the input containing a secret key and a plaintext (or ciphertext) should be stored in an input file. The secret key resides at the first line of the input file, with at most 1000 letters. The remaining content in the input file will be taken as the input plaintext (or ciphertext).

   The C program takes the first command line parameter as the option for encryption/decryption. It takes `-e` for encryption and `-d` for decryption. The name of the input file should be passed to your program as the second command line parameter:

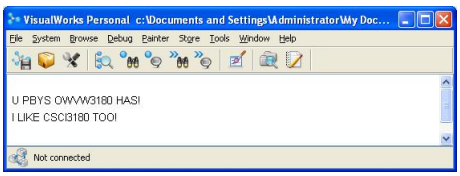
   ```
   ./VigenereCipher -e plaintext.txt
   ./VigenereCipher -d ciphertext.txt
   ```

   In the Smalltalk implementation, the input including the secret key, plaintext and ciphertext should be taken as parameters in the **Workspace** as shown in Figure 1a. You should run your program using our predefined statements in the Workspace.

3. **Output Specification**
   All the encrypted or decrypted letters should be output using **uppercase letters** and you should **ONLY** print the output ciphertext (or plaintext). In C implementation, results of the program should be printed to the **standard output**. In Smalltalk, the output should be printed to the **Transcript** window as shown in Figure 1b.



(a) Input statements in Workspace



(b) Output shown in Transcript window

Figure 1: Input and Output in Smalltalk implementation