Montgomery College
CMSC 203
Assignment 3 Design

Turn in pseudo-code for each of the methods specified in CryptoManager.java. Refer to the **Pseudocode Guideline** on how to write Pseudocode.

Turn in a test table with at least two tests for the Caesar Cipher and two for the Bellaso Cipher. Also, include at least one string that will fail because it has characters outside the acceptable ones.

Make sure your tests cover all the possible scenarios.

Input text	Input Key	Encrypted (method1)	Encrypted (method2)	Decrypt (method1)	Decrypt (method2)
HELLO	4	LIPPS		HELLO	
WORLD	5	\TWQI		WORLD	
NICO	BOY		PX\Q		NICO
man	BOY		Fail. Out of bounds		Fail. Out of bounds

Pseudocode Guideline

Pseudocode is code written for human understanding not a compiler. You can think of pseudocode as "English code," code that can be understood by anyone (not just a computer scientist). Pseudocode is not language specific, which means that given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

Pseudocode will be important to your future in Computer Science. Typically pseudocode is used to write a high-level outline of an algorithm.

As you may already know, an algorithm is a series of steps that a program takes to complete a specific task. The algorithms can get very complicated without a detailed plan, so writing pseudocode before actually coding will be very beneficial.

How to Write Pseudocode

There are no concrete rules that dictate how to write pseudocode, however, there are commonly accepted standards. A reader should be able to follow the pseudocode and hand-simulate (run through the code using paper and pencil) what is going to happen at each step. After writing pseudocode, you should be able to easily convert your pseudocode into any programming language you like.

We use indentation to delineate blocks of code, so it is clear which lines are inside of which method (function), loop, etc. Indentation is crucial to writing pseudocode. Java may not care if you don't indent inside your **if** statements, but a human reader would be completely lost without indentation cues.

Remember: Human comprehension is the whole point of pseudocode. So, what does pseudocode look like?

Finding the Fibonacci numbers till n:

Pseudocode	Real Code in Java

Declare an integer variable called n
Declare an integer variable sum.
Declare an integer variable f1
Declare an integer variable f2
If n is less than 2
sum =n
else
set sum to 0
set f1 and f2 to 1
repeat n times
sum = f1 + f2
f2 = f1
f1 = sum
end loop
print sum
	`int n,k, f1, f2, sum;`
	`if ( n < 2 )`
	`sum =n;`
	`else`
	`{`
	`sum=0;`
	`f1 = f2 = 1;`
	
	`for(k=2; k<n; k++)`
	`{`
	`sum = f1 + f2;`
	`f2 = f1;`
	`f1 = sum;`
	`}`
	`}`
	`System.out.println("Fibonacci of number " + n + " is "+ sum);`

Remember that pseudocode is not language specific so we are not looking for "almost Java" code, but instead, we are looking for a strong understanding of the algorithm at hand.

The user will enter the string to be tampered with, as well as the key.

When the cipher is selected, stringInBounds() will be executed

stringInBounds(String plainText): use a for loop to scan each individual character in the string to see if they are all within range (ASCII 32-95). If any one of those characters is not in those bounds, return false.

If the string is not in bounds, then the program will show an error message, and repeat the last two lines

If the user enters an integer, then the Ceasar cipher will execute

public static String encryptCaesar(String plainText, int key): Use a for loop to read each character in plainText and add the value of key to its ASCII value. If the resulting value goes over the limit, wrap around by subtracting the range (64) from that value to get the value that is needed. If that value is still

over the limit, repeat the last step until the final value is within range, and add that corresponding character to the encrypted String. Do this for every character and return the encrypted string when done.

If the decrypt button is selected:

public static String decryptCaesar(String encryptedText, int key): The same thing as encrypting, except the value of key is being subtracted from each character's ASCII value. If the resulting value is below the range, add the range value (64) to the value in question to get the final value. If it is still outside the rangem repeat the last step until it is not, and add that corresponding character to the decrypted string. Do this for every character in plainText and return the final result when done.

If the user enters something else and selects the encrypt button:

public static String encryptBellaso(String plainText, String bellasoStr): Use for loops to go through each character in each string. Add the ASCII values of each corresponding character. If the resulting value goes over the limit, wrap around by subtracting the range (64) from that value to get the value that is needed. If that value is still over the limit, repeat the last step until the final value is within range. Add that corresponding character to the encrypted String. Do this for every character and return the encrypted string when done.

If the user selects the decrypt button:

public static String decryptBellaso(String encryptedText, String bellasoStr): Use for loops to go through each character in each string. Subtract the ASCII value of the character from the key from its corresponding character's value in plainText. If the resulting value goes under the minimum range, wrap around by adding the range (64) to that value to get the value that is needed. If that value is still over the limit, repeat the last step until the final value is within range. Add that corresponding character to the encrypted String. Do this for every character and return the encrypted string when done.