**Montgomery College**

**CMSC 203**

**Assignment 3 Design**

Class: CMSC203 CRN 32293

Program: Assignment 3 Design

Instructor: Dr. Tanveer

Summary of Description: Create program that encrypts and decrypts text using Ceasar and Bellaso Ciphers with an integer or String key, respectively, between the ASCII values of 32 and 95. (“ ” & “\_”)

Due Date: 03/13/2023

Integrity Pledge: I pledge that I have completed the programming assignment independently.

I have not copied the code from a student or any source.

Student: Aryan Shrestha

**Part 1: Pseudo Code:**

Turn in pseudo-code for each of the methods specified in CryptoManager.java.   Refer to the [**Pseudocode Guideline**](#PSGdline)on how to write Pseudocode.

**isStringInBounds: For a declared String named text…**

* **Declare boolean value within and set equal to true**
* **For any character in inputted text, set within to false**
* **Return within**

**rangeInBetween: For a declared character named Btwn…**

* **If Btwn is in between 32 and 95 in ASCII table (‘ ‘ and ‘\_’; space and underscore), return true**
* **If not, return false**

**caesarEncryption: for declared string caesarText and integer key…**

* **Declare String variable encryptedCaesar equal to “”;**
* **If isStringBounds with respect to caesarText**
  + **Then: for any character in caesarText,**
    - **Declare char variable ‘character’ equal to caesarText @ int i**
    - **Declare integer encryptedASCII = to ASCII value of ‘character’ + Cyber Key**
      * **NTS: To do so, let ‘character’ temporarily be integer by (int)character**
    - **While encryptedASCII is above UPPER\_RANGE, subtract RANGE from it.**
    - **Add characterized encryptedASCII to encryptedCaesar**
* **Else**
  + **Set encryptedCaesar to given error message in CryptoManagaerTestPublic.java**
* **Return encryptedCaesar**

**bellasoEncryption: for declared string bellasoText and String\* (not integer like before) key…**

* **Declare String variable encryptedBellaso equal to “”;**
* **Declare integer bellasoLength = length of key**
* **For any character in bellasoText**
  + **Declare char variable ‘character’ equal to bellasoText @ int i**
  + **Declare integer encryptedASCII = to ASCII value of ‘character’ + Cyber Key**
    - **NTS: To do so, let ‘character’ temporarily be integer by (int)character**
  + **While encryptedASCII is above UPPER\_RANGE, subtract RANGE from it.**
  + **Add characterized encryptedASCII to encryptedBellaso**
* **Return encryptedBellaso**

**caesarDecryption: for declared string encryptedCaesar and integer key…**

* **Declare String variable decryptedCaesar equal to “”;**
* **For any character in encryptedCaesar,**
  + **Declare char variable ‘character’ equal to encryptedCaesar @ int i**
  + **Declare integer decryptedASCII = to ASCII value of ‘character’ + Cyber Key**
    - **NTS: To do so, let ‘character’ temporarily be integer by (int)character**
  + **While decryptedASCII is above UPPER\_RANGE, subtract RANGE from it.**
* **Add characterized decryptedASCII to decryptedCaesar**
* **Print programmer name**
* **Return decryptedCaesar**

**bellasoDecryption: for declared string encryptedBellaso and integer key…**

* **Declare String variable decryptedBellaso equal to “”;**
* **For any character in encryptedBellaso,**
  + **Declare char variable ‘character’ equal to encryptedBellaso @ int i**
  + **Declare integer decryptedASCII = to ASCII value of ‘character’ + Cyber Key**
    - **NTS: To do so, let ‘character’ temporarily be integer by (int)character**
  + **While decryptedASCII is above UPPER\_RANGE, subtract RANGE from it.**
* **Add characterized decryptedASCII to decryptedBellaso**
* **Print programmer name**
* **Return decryptedBellaso**

**Part 2: Comprehensive Test Plan**

Turn in a Test Plan table. Test Plan should include:

* at least two tests for the Caesar Cipher
* at least two for the Bellaso Cipher.
* at least one string that will fail because it has characters outside the acceptable ones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input text | Input Key | Encrypted (method1)  caesar | Encrypted (method2)  bellaso | Decrypt (method1)  caesar | Decrypt (method2)  bellaso |
| HOTEL CALIFORNIA | 3 | KRWHO#FDOLIRUQLD |  | HOTEL CALIFORNIA |  |
| {THIS~IS GOING TO~FAIL} | FAILURE |  | AUQU(PNY!P[^ L&UXJ{SNR> |  | ;THIS>IS GOING TO>FAIL= |
| GRAPE FANTA | 5 | LWFUJ%KFSYF |  | GRAPE FANTA |  |
| 1@3$5^7\*9+[]:’,>/? | RANDOM |  | CAA(D+I+G/\*\*(:B>L |  | 1@3$5^7\*9+[]:’,>/? |

**Make sure your tests cover all the possible scenarios.**

**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?

|  |  |
| --- | --- |
| **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); |

**Finding the Fibonacci numbers till n:**

**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.**