**Montgomery College**

**CMSC 203**

**Assignment 3 Design**

Class: CMSC203 CRN XXXX

 Program: Assignment 3 Design

Instructor:

 Summary of Description: (Give a brief description for a Program)

 Due Date: 03/14/2021 (<03/12/2021>)

 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: Gregory Guyah

**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.

* **Declare constant string for lower bound, upper bound, range**
* **Allowed bounds of ACII codes**
* **Constant string Boolean string plain text bounds**
* **Plaintext string encrypted if within allowed bounds**
* **Int index**
* **False boundary**
* **Return boundary**
* **Encrypt string to the Caesar Cipher**
* **Integer key specifies offset**
* **Each character plaintext is replaced by character**
* **Return encrypted string**
* **Encrypt string to the Bellaso Cipher**
* **Integer key specifies offset**
* **Each character plaintext is replaced by character**
* **Return encrypted string**
* **Decrypts string according to Caesar Cipher**
* **Encrypted text to be decrypted**
* **Return plain text string**
* **Decrypts string according to Bellaso Cipher**
* **Encrypted text to be decrypted**
* **Return plain text string**

**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) | Encrypted (method2) | Decrypt (method1) | Decrypt (method2) |
|  | 15 | Caesar Cipher | Bellaso Cipher | Caesar Cipher | Bellaso Cipher |
|  | 15 | Bellaso Cipher | Caesar Cipher | Bellaso Cipher | Caesar Cipher |
|  | HELLO | Caesar Cipher | Bellaso Cipher | Caesar Cipher | Bellaso Cipher |
|  | HELLO | Bellaso Cipher | Caesar Cipher | Bellaso Cipher | Caesar Cipher |

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

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Chart, scatter chart

Description automatically generated

* Learning Experience: highlight your lessons learned and learning experience from working on this project.
* What have you learned? I learned how to run multiple test
* What did you struggle with? Struggled with creating my pseudo-code
* What will you do differently on your next project? Get better at making a pseudo-code
* Include what parts of the project you were successful at, and what parts (if any) you were not successful at. Making the program run correctly.

Assignment 3 Check List

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment#\_Moss.zip | **<Yes or No>** | **Yes** |
|  | * FirstInitialLastName\_Assignment#.docx/.pdf | **<Yes or No>** | **Yes** |
|  | * Source java files | **<Yes or No>** | **Yes** |
|  | **Program compiles** | **<Yes or No>** | **Yes** |
|  | **Program runs with desired outputs related to a Test Plan** | **<Yes or No>** |  |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **<Yes or No>** | **Yes** |
|  | * Screenshots for each Test case listed in the Test Plan | **<Yes or No>** | **Yes** |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **<Yes or No or N/A>** | **Yes** |
|  | * UML Diagram (if required) | **<Yes or No or N/A>** | **N/A** |
|  | * Algorithms/Pseudocode (if required) | **<Yes or No or N/A>** | **Yes** |
|  | * Flowchart (if required) | **<Yes or No or N/A>** | **N/A** |
|  | * Lessons Learned | **<Yes or No>** | **Yes** |
|  | * Checklist is completed and included in the Documentation | **<Yes or No>** | Yes |

**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.**