**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**](#PSGdline)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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input text | Input Key | Encrypted (method1) | Encrypted (method2) | Decrypt (method1) | Decrypt (method2) |
| Same as example | Same as example | Same as example | Same as example | Same as example | Same as example |
| I enjoy Computer Science | 15 | Encrypt a String | - | Decrypt a String | - |
| Blue is my favorite color | 15 | Encrypt a String | - | Decrypt a String | - |
|  |  |  |  |  |  |

**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.**

**Pseudocode:**

1. **Start**
2. **Data Manager Class – CryptoManager.java**
3. **Create a Boolean method named isStringInBound with 1 parameter called, String plainText.**
4. **Get ASCII code for all characters.**
5. **This method will determine if string is within allowable bounds of ASCII codes according to the LOWER\_RANGE & UPPER\_RANGE.**
6. **Return plainText as true if all characters are within range.**
7. **Return plainText as false if any character is outside range.**
8. **Create a String method named caeserEncryption with 2 parameters: String plaintext & integer key.**
9. **Get ASCII code of the character.**
10. **return encrypted String.**
11. **If plainText is not in bounds, return “The selected string is not in bounds”.**
12. **Create another method named caeserDecryption with 2 parameters: String encryptedText & integer keys.**
13. **Compare characters with ASCII code to decrypt the text.**
14. **Return original plainText.**
15. **Create another method named bellasoDecryption with 2 parameters: String plainText & String bellasoStr.**
16. **Return encrypted String.**
17. **Create another method named bellasoDecryption with 2 parameters: String encryptedText & String bellasoStr.**
18. **This method is inverse of bellasoDecryption.**
19. **Return original plain String.**
20. **GUI Driver Class**
21. **JUnit Test**
22. **Stop**