## Final Project: Deliverable 3, Instructions

Consider the Matyas-Meyer-Oseas (MMO) hash function presented in the textbook1, and also in the Module 7 notes2. You should implement a 128-bit version of this hash function.

Recall that the input ***x*** to the hash function has arbitrary length, the key ***k*** for the hash function is an odd integer between 1 and 2128-1, and that we encrypt a 128-bit block via multiplication by ***k*** modulo 2128. Finally, recall that ***k*** is replaced by the function ***g*** that derives the key.

We will use a modified version of **g** as defined in the Module 7 lecture notes. Specifically, our modified **g(s)** function is defined as follows:

* If the last bit in the string **s** is 1, then g returns the same string **s**.
* If the last bit in the string **s** is 0, then g makes the last bit a 1 and returns this modified version of **s**.

Finally, the MMO hash function requires an initial value of **H0**. You may use a random value for **H0**, and you must clearly call this variable **H0**in your initialization code. I will change this value when I test your code so I should be able to locate it without difficulty.

**Assume the inputs to your code are a plaintext file name, and a hash value file name. You are given a stubbed-out file MMO.py that you must complete, following the commented instructions provided in it. Almost every method requires you to complete all or some of it. There are also helper functions you can use in the StringConversion.py file. Note that the inputs are expected as ASCII strings; you can use methods from StringConversion.py to convert them to binary “strings”.**

Specifically, you must provide the following:

1. The method MMO\_hash(**plaintextFileName**, **hashedValueFileName**) that hashes the alphabetic text in the file named by the first parameter into the 128-bit message digest that is written to the file named in the second parameter.
2. Two test cases for your system and instructions for how to get these test cases to pass on your codebase.
3. Finally, review the attached rubric for this deliverable, and for each performance criterion, please self-evaluate your work by describing whether you completely or partially met expectations, or did not meet expectations, on this criterion.

## Format

Please submit your answers:

* To items 1 and 2 as a zip file containing the complete codebase for your implementation, including test cases and a readme file with instructions on how to run your code, and its test cases.
* To item 5 as a document providing the required self-assessment.

Please keep in mind:

* Your code should run correctly and be implemented efficiently.
* Proper indentation and variable naming should be followed. Use meaningful variable names.
* Use methods as appropriate.
* Discussion should address the specific questions asked.

*1. Understanding Cryptography, Christof Paar and Jan Pelzl, Chapter 11, page 305*

*2.* CYB 710: Module 7 Notes