READ THIS FIRST:

Do your best to do every item on your own; if you cannot immediately do an item, go on to others and then come back to it later. Please ask your professor whenever you have a question. **Due: Tuesday, March 29, 2016.**

Name: _____

Goals:

- To get you familiar with one of the best cryptosystems of our times.
- To get you ready for the next lab since we will continue programming parts of AES.
- Practice getting around in and using GitHub.

Background: It is public domain information that the U.S. Government allows usage of the Advanced Encryption Standard (AES) to protect SECRET and TOP SECRET information depending on the key-length used. You have already developed the part of AES that produces keys for every round of encryption. See Lab 2 for details. This laboratory assignment builds on Lab 2 and continues the development of the functions of AES.

Instructions: You will use your previously created AES files that included: `driver.java` and `aescipher.java`. But this time, let us all change the name and have standard class name and files as follows: `Driver.java` and `AEScipher.java`. In your `AEScipher.java` file, you will add the following new methods:

1. *Method for AES Add Key.* Write a method with syntax `outStateHex = aesStateXOR(sHex, keyHex)` whose inputs and output are four by four matrices where every element is a pair of hex digits and that will perform the "Add Round Key" operation; that is, the entries of the output matrix are simply the XOR of the corresponding input matrix entries. Here is a test case:

$$\begin{bmatrix} 54 & 4F & 4E & 20 \\ 77 & 6E & 69 & 54 \\ 6F & 65 & 6E & 77 \\ 20 & 20 & 65 & 6F \end{bmatrix} \oplus \begin{bmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{bmatrix} = \begin{bmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{bmatrix} \tag{1}$$

2. *Method for AES Nibble Substitution.* Write a method with syntax `outStateHex = aesNibbleSub(inStateHex)` whose inputs and out are 4 by 4 matrices of paris of hex digits, and that will perform the "Substitution" operation, i.e., the entries of the output matrix result from running the corresponding input matrix entries through the AES S-Box. Hint: you should use the method you created in Lab 2 `outHex =`

`aesSBox(inHex)` somehow in this method. Here is a test case:

$$\text{this input } \begin{bmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{bmatrix} \text{ produces the following output } \begin{bmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{bmatrix} \quad (2)$$

3. *Method for AES Shift Rows.* Write a method with syntax `outStateHex = aesShiftRow(inStateHex)` whose inputs and output are 4 by 4 matrices of pairs of hex digits ant will perform the Shift Row operation of the AES to transform the input state matrix into output state. Here is a test case:

$$\text{this input } \begin{bmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{bmatrix} \text{ produces the following output } \begin{bmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{bmatrix} \quad (3)$$

4. *Method for AES Mix Column.* Write a method with syntax `outStateHex = aesMixColumn(inStateHex)` whose inputs and output are 4 by 4 matrices of pairs of hex digits and will perform the Mix Column operation of AES to transform the input state into output state. Here is a test case:

$$\text{this input } \begin{bmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{bmatrix} \text{ produces the following output } \begin{bmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{bmatrix} \quad (4)$$

   This one is a little tricky because it is doing miltiplications in the Galois field. So, this item requires from you to do some graduate level research. But do not worry, I have provided great resources below in the -resources- section.

5. *Method for AES Encryption.* Write a method with syntax `cTextHex = aes(pTextHex, keyHex)` that will perform AES encryption following the algorithm we discussed in class and shown in Figure 1. Here is a test case; for the following key:

   ```
   5468617473206D79204B756E67204675
   ```

   and the following plaintext:

   ```
   54776F204F6E65204E696E652054776F
   ```

   the output should be:

   ```
   29C3505F571420F6402299B31A02D73A
   ```

   Intuitively, this method will make use of all the methods you have previously developed, so, make sure everything is properly tested.

Along with your `.java` files, you must also submit at least three test cases, named `test.1.txt`, `test.2.txt`, and `test.3.txt`. These test cases must be different from the ones found commonly on-line. You can (and should) use test vectors from the web, but you will then create your own and submit those.

Your `Driver.java` program will test your implementation by calling `aes()` providing valid data. Your driver should read the system key, K_e, and plaintext, p, from standard input, i.e., `System.in`, using file

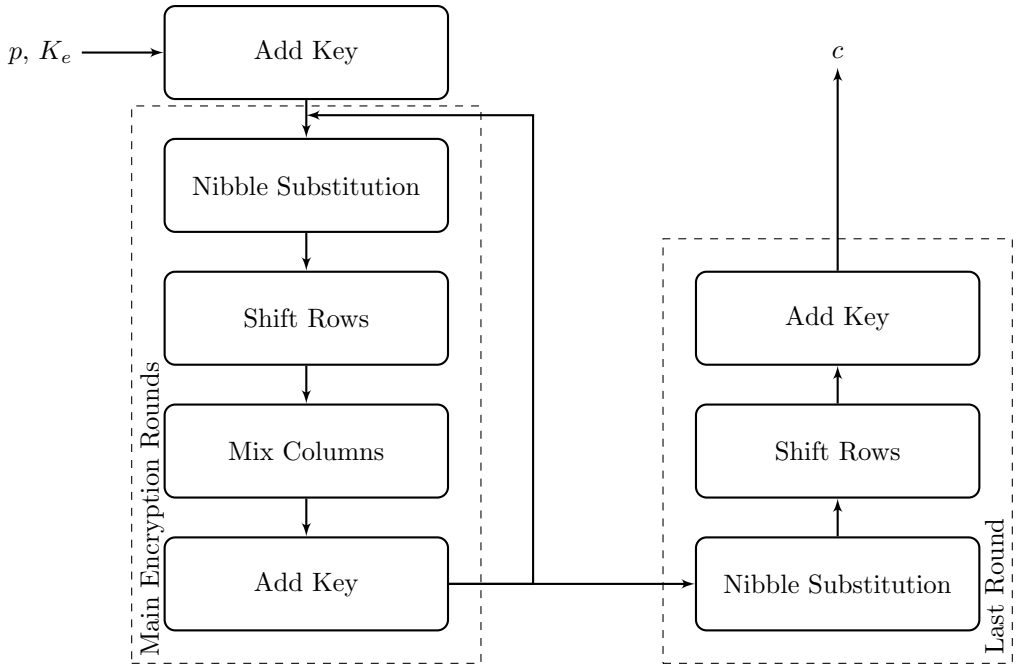

Figure 1: AES sequence diagram.

redirection from a text file, and will print the result to standard output, i.e., `System.out`. Your program **must run from command line** as follows:

```
> java Driver < testCase.txt
29C3505F571420F6402299B31A02D73A
```

In this example the contents of the `testCase.txt` are only the following two lines:

```
5468617473206D79204B756E67204675
54776F204F6E65204E696E652054776F
```

that is, the file only has two lines; the first is the key, and the second is the plaintext. Both have sizes of 128-bits. Your program **must only accept data in this format**. E.g., if you create other test cases, which is highly recommended, make sure all test cases have the same format as this test case file.

Please do not forget that one **mandatory** requirement of this assignment is that you follow the style guidelines in `http://www.reev.us/cmpt220f15/style.html`

Evaluation: You will be evaluated under the following criteria

- Does if follow the specified coding style guidelines `http://www.reev.us/cmpt220f15/style.html`?
- Does it produce correct output? (Is the plaintext correctly encrypted?)
- Does it produce output in the correct format? (Does it look -exactly- as shown in the example?)
- Is all the code well-documented? (Does it have enough comments?)
- Is the code efficient? (Is the progammer making an effort to produce efficient code?)
- Are the test cases good? (Do they test boundaries or special patterns?)

Resources:

- Chapter 3 of the textbook
- The Java coding style guidelines: `http://www.reev.us/cmpt220f15/style.html`
- General info about AES `https://en.wikipedia.org/wiki/Advanced_Encryption_Standard`
- General info about AES S-box: `https://en.wikipedia.org/wiki/Rijndael_S-box`
- General info about AES mixing columns: `https://en.wikipedia.org/wiki/Rijndael_mix_columns`
- A tutorial on AES mixing columns: `http://www.angelfire.com/biz7/atleast/mix_columns.pdf`
- A few test vectors: `http://www.inconteam.com/software-development/41-encryption/55-aes-test-vectors`

Submission: Submit the following files

1. `Driver.java`
2. `AEScipher.java`
3. `test.1.txt`
4. `test.2.txt`
5. `test.3.txt`

to your GitHub repository **before** due date (see the top of this document). Remember to include your name, the date, and the assignment in the (copious, meaningful, and accurate) check-in messages. Commit often!