SY201 Name(s): ____________________

Alpha(s): ____________________

Assignment Type:	Lab	Collaboration Policy:	Default
Assignment Title:	Lab 4 Directions	Submit Project Name:	lab4
Electronic submission due: 2359 on 1 Oct Paper submission due: start of class on 2 Oct Submission instructions: http://courses.cyber.usna.edu/SY201/calendar.php?load=policy			

1. Assignment Overview

In this assignment you will create a program that performs stretching, repeating a hashing algorithm a number of times. Modern-day stretching is more complex and secure than the basic stretching that you will implement in this lab, but the core concepts are the same. The hash functions that you will use in this lab are from the same family of hash functions that are used in practice, and the number of rounds that you will implement section are the same number of rounds used in practice. MD5 (Apache web servers .htpasswd), SHA-1 (WPA2), and SHA-256 (Apple FileVault2) use 1,000, 4,096, and 41,000 rounds, respectively

2. Background Research

a. Read the SY110 course notes on *Hashing*: https://www.usna.edu/CyberDept/sy110/calendar.php?type=lab&event=6. Answer question 1 on the worksheet.

b. Read the *Salt (cryptography)* Wikipedia article available at this link: https://en.wikipedia.org/wiki/Salt_(cryptography). Answer questions 2 and 3 on the worksheet.

c. Read the *Key stretching* Wikipedia article, available at https://en.wikipedia.org/wiki/Key_stretching. Answer questions 4 and 5 on the worksheet.

d. Like most programming languages Python allows programmers to reuse code, i.e. incorporate source code that other developers wrote into new programs. Python provides this functionality by importing modules; module(s) is the term Python uses, other languages call them libraries. In Python the import word is reserved to import a Python module. Enter these commands into the Python interpreter and read the results.

i. import hashlib # import the hashing library module
ii. help(hashlib) # help information for the hash library module

e. Importing modules is especially useful when dealing with cryptographic operations as any mistakes made in writing your own crypto can disable any cryptographic security benefit. But while the mathematics behind cryptography is well beyond the scope of SY201, you will see it later in the major.

f. Because the hashlib module is extensive, we will give an example of how you will use it in this lab:

```
hashValue = hashlib.md5( bytes( "spam", "ASCII" ) ).hexdigest()
```

This line of code computes the hexadecimal representation of the hash of the string *spam* and stores it in the variable hashValue. It uses the hash function md5 but other hash functions could be used (e.g., sha1 and sha256).

3. Specification - create a file lab4.py with the following functions and **includes comments adjacent to each function that describe the function's purpose and use**:

a. Write your alpha and section number in a comment at the top of your program

b. Write a function named compute_hash that:
   i. Takes three arguments: a string (a string to be hashed), an integer (number of hash rounds), and a string (the hash function to be used: 'MD5', 'SHA1', or 'SHA256')
   ii. Returns the hexadecimal representation of result of conducting the passed number of hashing rounds on the passed string

c. Write a function named get_hash_function that:
   i. Takes no arguments
   ii. Precisely prints out the following:

```
Hash Algorithms
(1) MD5
(2) SHA-1
(3) SHA-256
```

   iii. Prompts the user for a numerical selection using "Selection: "
   iv. Returns one of the strings "MD5", "SHA1", or "SHA256" based upon the user's input (the number 1, the number 2, or the number 3)

d. Write a function named get_hash_rounds that:
   i. Takes no arguments

ii. Prompts the user for the number of hash rounds using: “Rounds: ”
iii. Returns the integer value of the user’s input

e. Write a function named main that (in this order):
   i. Takes no arguments
   ii. Precisely prints out the following:

```
Hash Algorithms
(1) MD5 1000 Rounds
(2) SHA-1 4096 Rounds
(3) SHA-256 41000 Rounds
(4) User Specified
(5) Quit
```

   iii. Prompts the user for a numerical selection using “Selection: ”
   iv. If the user chooses option five (user input: 5), your program should ask for no additional input and generate no additional output
   v. If the user chooses option four (user input: 4), your program should retrieve the user’s choice of hash function and number of hash rounds (in that order) using the functions: get_hash_function and get_hash_rounds
   vi. If the user chooses an option between one and four(user input: 1-4), your program should:
      1. Prompt the user for a password using “Password: ”
      2. Prompt the user for a salt using “Salt: ”
      3. Compute the number of rounds of the chosen hash function on the salted password
      4. Print out the result as “[HASH_FUNCTION] - [HASH_ROUNDS] Rounds - [RESULT]”

4. <u>Test the Program</u>

Run your program and call the main function. Try all variations of the menu options. Answer question 6 on the worksheet.

5. <u>Block Diagram</u>

Draw a block diagram on your worksheet (number 7) that describes the control flow and decision-making of the function main the other functions that it calls.