Homework 9, Due Date*: 11:59pm 12/06/2019 (Friday), Cutoff Date*: 11:59pm 12/08/2019 (Sunday)

****Submission will ABSOLUTELY NOT be accepted after the cutoff deadline**

20 Extra Bonus Points: Server is able to simultaneously support ***multiple*** clients. All the requirements that are highlighted in green in Task I are for extra bonus points only. Please annotate it in your submission comments and a README.txt file on cs3750a.

NO peer/team programming is allowed. You must work on this assignment individually without sharing code with anyone. You are very welcome to use and modify the example code for Echo client and server.

Grading: Your programs will be graded via testing and points are associated with how much task it can complete. A program that cannot be compiled or crashes while running will receive up to 5% of the total points. A submission of .java files that are similar to any online Java program with only variable name changes will receive 0% of the total points.

Programming in Java is required, since this topic is all about the use of PKCS12 or JKS keystore as well as SSL programing via Java Secure Socket Extension (JSSE). NO other programming language may be used for this assignment.

Submission:
1) upload your two .java file(s) on Blackboard (email is NOT accepted)
2) in the "submission comments" on Blackboard, list the names and passwords of your keystore and truststore, and the alias and password of your keys.
3) set up the server program (.java and .class files) on cs3750a under HW09/server together with the keystore
4) set up the client program (.java and .class files) on cs3750b under HW09/client together with the truststore
5) in a file named "keyinfo.txt" under HW09/server on cs3750a, list the names and passwords of your keystore and truststore, and the alias and password of your PrivateKeyEntry and trustedCertEntry.
6) the file named "clientOutputs.txt" under HW09/client on cs3750b (see task II for more details).

Recommended References: You are highly recommended to complete Lab 03 before working on this assignment.

[1] "**Internet Security**" **PPT slids** at "Blackboard→Content→PPT Slides": Slide 2 "Secure Sockets Layer (SSL)" through Slide 17 "SSL Programming: Dynamically Set System Properties"

[2] **Lab manual** for Lab3 SSL at "Blackboard→Content→Labs"

[4] **"Echo Server and Client"** and "TCP ToUpperCase with Multithreaded Server" at "Blackboard→Content→Security Programming in Java/C"

[5] **JSSE Reference Guide** https://docs.oracle.com/javase/9/security/java-secure-socket-extension-jsse-reference-guide.htm#JSSEC-GUID-93DEEE16-0B70-40E5-BBE7-55C3FD432345

Task I (85%): Write a client Java program and a server Java program to implement the following secure Info Collection protocol on top of **SSL/TLS** service. TCP or UDP service ***cannot*** be used directly in your programs, i.e., any TCP/UDP class such as Socket and ServerSocket ***cannot*** be used in your source code although the ssl sockets used in your source code will certainly use the TCP service.

- **InfoCollection Server** Program:
  1. Take **one argument, which specifies the port number that the server listens to,** and create the SSL Server Socket.
  2. Listen to the given port and wait for a connection request from an InfoCollection Client.
  3. Create a new thread and an SSL socket for every incoming SSL connection from an InfoCollection client. While the original thread goes back to Step 2, the new thread continues with the following steps.
  4. Get the **session** of this SSL socket and display the following information (hint: use the **get…()** methods of this session.)
     ```
     Peer host is ……
     Cypher suite is ……
     Protocol is ……
     Session ID is ……
     The creation time of this session is ……
     The last accessed time of this session is ……
     ```
  5. Send questions to and collect information from the InfoCollection client
     a. Send "`User Name: `" to client, read the response from client, create a txt file named as <this user ID>.txt, and write "`User Name: <response from client>`" as the first line in this file.
     b. Send "`Full Name: `" to client, read the response from client, and add a line, "`Full Name: <response from client>`", to the file created in Step 5.a.
     c. Send "`Address: `" to client, read the response from client, and add a line, "`Address: <response from client>`", to the file created in Step 5.a.
     d. Send "`Phone number: `" to client, read the response from client, and add a line, "`Phone number: <response from client>`", to the file created in Step 5.a.

e. Send "`Email address: `" to client, read the response from client, and add a line, "`Email address: <response from client>`", to the file created in Step 5.a.
f. Close the file created in Step 5.a.
g. Send "`Add more users? (yes or any for no)`" to client, read the response from client. If the client responds "`yes`", go back to Step 5.a and repeat. Otherwise, close the SSL connection and terminate this new thread (and loop back to Step 2 if the server is single-threaded).

- **InfoCollection Client** Program:
  1. Take **two arguments, which specify the *ip/dns* and the *port* number of the remote InfoCollection Server.**
  2. Create a SSL socket to connect to your InfoCollection server. Catch the exception, terminate the program, and display error messages on the standard output if any.
  3. Get the session of this SSL socket and display all the information as what is listed in Server program's Step 4.
  4. Read ***each*** question from the Info Collection server, one at a time, and display each question on the standard output to ask the user to input the answer. Read the user's answer from the standard input and send it to the server.
     a. If the user answers `any except for` "`yes`" to the InfoCollection Server's Step 5.g, close the SSL connection and terminate this program AFTER sending `the user's answer` to the InfoCollection Server.
     b. Otherwise, repeat Step 4.

Task II (15%): Generate keys and test your program on **cs3750a** and **cs3750b**.

Warning: to complete this part, especially when you work at home, you must first (1) **connect to the MSUDenver VPN** via **GlobalProtect**; then (2) **connect to** the virtual servers **cs3750a.msudenver.edu** and **cs3750b.msudenver.edu** using ***sftp*** and ***ssh*** command on MAC/Linux or ***PUTTY*** and ***PSFTP*** on Windows. For details, you may refer to **Lab 1.**

The server program always has to start BEFORE the client program in your test.

1. Create a directory "**HW09**" under your home directory on cs3750a and cs3750b. Create a subdirectory "**server**" on **cs3750a** for the ***server*** program and your ***keystore***. Make a subdirectory "**client**" on **cs3750b** for the ***client*** program and your ***truststore*** there.
2. Use ***keytool*** to create a simple JKS keystore suitable for use with JSSE. Make a PrivateKeyEntry in your keystore in **HW09/server**, then make a corresponding trustedCertEntry in your truststore in **HW09/client**.
3. In a file named "keyinfo.txt" under **HW09/server on cs3750a**, list the names and passwords of your keystore and truststore, and the alias and password of your PrivateKeyEntry and trustedCertEntry.
4. RUN and TEST your **server** program in **HW09/server** on cs3750a. You must use the ***port number*** assigned to you in the file named "***portAssignment.pdf***", which is posted Blackboard under "Lab03 SSL", in your **server** program as the local port number to create a SSL Server Socket, which will wait for a connection request from any client.
5. RUN and TEST your **client** program in **HW09/client** on cs3750b, copy & paste the outputs of this client program during your test to a file named "***clientOutputs.txt***". (Hint: in your **client** program, your need to use **cs3750a.msudenver.edu** as the *remote DNS/IP* and the **port number** assigned to you as the *remote port number* to create a SSL socket that is connected to your server program.)
6. (This step can be done simultaneously with the above step if your server program supports multithreading.) RUN and TEST your **client** program on your local computer, while the **server** program is running on cs3750a. You may either copy the truststore to your local computer or import the key certificate to a local truststore on your local computer.