**Workshop 9: Multi-Threading**

In this workshop, you backup an encrypted text file of characters to a binary file, restore the encrypted data from the binary file and add multi-threading to the solution.

**Learning Outcomes**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* process partitioned data on two or more threads
* write a set of characters to a file in binary mode
* read a set of characters from a file in binary mode
* bind a function to its arguments

**Submission Policy**

The *in-lab* section is to be completed during your assigned lab section. It is to be completed and submitted by the end of the workshop period. If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the *in-lab* portion after the period. If you do not attend the workshop, you can submit the *in-lab* section along with your *at-home* section (see penalties below). **In order to get credit for the *in-lab* portion, you must be present in the lab for the entire duration of the lab.**

The *at-home* portion of the workshop is due on the day that is four days after your scheduled *in-lab* workshop (@ 23:59:59), **even if that day is a holiday**.

All your work (all the files you create or modify) must contain your name, Seneca email, student number and the date of completion (use the following template):

// Name:

// Seneca Student ID:

// Seneca email:

// Date of completion:

//

// I confirm that the content of this file is created by me,

// with the exception of the parts provided to me by my professor.

You are responsible to back up your work regularly.

**Late Submission Penalties**

The workshop can be submitted up to **1 (one) day** late (the day that is 5 days after the lab period); submissions received on this day are considered **late** and are subject to penalties:

* only *in-lab* portion submitted late (after the end of the lab period): 0 for *in-lab* portion, max 7/10 for the entire workshop.
* only *at-home* portion submitted late (more than 4 days after the lab period): max 4 for the *at-home* portion, max 7/10 for the entire workshop.
* both *in-lab* **and** *at-home* portions submitted late: max 4/10 for the entire workshop.
* when the submission closes, if the workshop is not complete, the mark for the entire workshop will be 0/10. The workshop is considered complete if there are two separate submissions (*in-lab* submission and *at-home* submission) containing the *in-lab code*, *at-home code* and *reflection*.

The submission is considered closed at the end of the day that is 5 (five) days after the lab period.

***In-Lab***

The three source files provided for this workshop are

* SecureData.h -- already implemented, **do not modify**
* SecureData.cpp -- partially implemented
* w9.cpp -- already implemented, **do not modify**

The SecureData class holds text in encoded form along with the number of bytes in the text including the null terminator.

The Cryptor function object holds the encryption/decryption logic used by the SecureData class. This object is fully implemented.

Your tasks for this part of the workshop are to complete the backup() and restore() member functions of the SecureData class. Your code in each case should open the binary file in the appropriate mode and throw an exception if opening fails.

For backup, your code simply writes the data to the file.

For restoration, your code deallocates the existing data, determines the number of bytes to be read, allocates memory for them, and reads the data from the file.

**w9 Module (supplied)**

The tester module for the in-lab portion has been supplied. **Do not modify the existing code!**

When doing the workshop, you are encouraged to write your own tests, focusing on a single implemented feature at the time, until you get all functionality in place.

**Sample Output**

When the program is started with the command (the input file is provided):

w9.exe input.txt encoded.dat.txt K

the output should look like the one from the sample\_output.txt file.

**⚠️Important:** The output file encoded.dat is not automatically checked; before submission make sure that the file contains binary encoded data (and not the original text). Submissions that do not have encoded data in encoded.dat will be rejected.

**Submission (30%)**

To test and demonstrate execution of your program use the same data as shown in the output example.

Upload your source code to your matrix account. Compile and run your code using the latest version of the g++ compiler (available at /usr/local/gcc/9.1.0/bin/g++) and make sure that everything works properly.

Then, run the following command from your account (replace profname.proflastname with your professor’s Seneca userid):

~profname.proflastname/submit 305XXX\_w9\_lab

and follow the instructions. Replace XXX with the section letter(s) specified by your instructor.

**⚠️Important:** Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

***At-Home***

Your second task is to multi-thread the conversion in the code() member function of the SecureData class.

Select a reasonable number of threads and partition the data equally amongst all of the threads. Use std::bind from the functional library to bind the converter.

**w9 Module (supplied)**

The tester module for the in-lab portion has been supplied. **Do not modify the existing code!**

When doing the workshop, you are encouraged to write your own tests, focusing on a single implemented feature at the time, until you get all functionality in place.

**Sample Output**

When the program is started with the command (the input file is provided):

w9.exe input.txt encoded.dat.txt K

the output should look like the one from the sample\_output.txt file.

**⚠️Important:** The output file encoded.dat is not automatically checked; before submission make sure that the file contains binary encoded data (and not the original text). Submissions that do not have encoded data in encoded.dat will be rejected.

**Reflection**

Study your final solution, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. **This should take no less than 30 minutes of your time.**

Create a **text** file named reflect.txt that contains your detailed description of the topics that you have learned in completing this particular workshop and mention any issues that caused you difficulty and how you solved them. Include in your explanation—**but do not limit it to**—the following points:

* why was necessary to use binary files in this worksho and how did you use them
* why would be important to bind a function to its arguments, and how was it useful in this workshop
* the advantages of using multiple threads, and how did you accomplished multi-threading in this workshop

To avoid deductions, refer to code in your solution as examples to support your explanations.

**Quiz Reflection**

Add a section to reflect.txt called **Quiz X Reflection**. Replace the **X** with the number of the last quiz that you received and list all questions that you answered incorrectly.

Then for each incorrectly answered question write your mistake and the correct answer to that question. If you have missed the last quiz, then write all the questions and their answers.

**Submission (30% for code, 40% for reflection)**

To test and demonstrate execution of your program use the same data as shown in the output example above.

Upload the source code and the reflection file to your matrix account. Compile and run your code using the latest version of the g++ compiler (available at /usr/local/gcc/9.1.0/bin/g++) and make sure that everything works properly.

Then, run the following command from your account (replace profname.proflastname with your professor’s Seneca userid):

~profname.proflastname/submit 305XXX\_w9\_home

and follow the instructions. Replace XXX with the section letter(s) specified by your instructor.

**⚠️Important:** Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.