MiniProject 2 (Smart pointers)

Background

Check out the smart pointers in the standard library, e.g.:

- http://umich.edu/~eecs381/handouts/C++11_smart_ptrs.pdf (also commented in OMA)
- http://www.informit.com/articles/article.aspx?p=2085179 (you should try the example code snippets)
- https://www.codeproject.com/Articles/15351/Implementing-a-simple-smart-pointer-in-c (smart pointer class example implementation)

You can try `shared_ptr` as follows:
Create a simple class whose constructor and destructor print something. Give one object of the class to be taken care of by `shared_ptr`. Pass `shared_ptr` as a parameter to some function, return `shared_ptr` as the return value of the function, and make placements between `shared_ptrs`. Pay attention to the point at which the object is destroyed. What is the reference count at what point in the program?

Part 1 Basic Log class (1 point)

Design and implement the class Log_Ptr. The class is given to take care of another entity, which is reserved from the heap. Lines are written to a log file (e.g. to the file "log_ptr.txt" or directly to the console).

- Log_Ptr must have the following properties (write a test program):
- Log_Ptr maintains a pointer to the referenced entity. The object can be of any type (template-class).
- Log_Ptr is given the referenced entity as a constructor parameter. The constructor writes the corresponding line in the log file: <time stamp> ownership transferred <memory address of referenced entity>.
- The referenced entity is destroyed in the destructor of the Log_Ptr. The destructor writes the corresponding line in the log file:
- <time stamp> object destroyed <memory address of referenced object>
- The Log_Ptr assignment operator and copy constructor are disabled (hint: https://www.geeksforgeeks.org/explicitly-defaulted-deleted-functions-c-11/)

Part 2 (Arrow and derefence operators) (1 point)

Add the following properties to the Log_Ptr class:

- Add to the class -> an operator to access the referenced object. The operator writes the corresponding line in the log file: <time stamp> operator-> <memory address of referenced entity>.

- Add to class * the operator to access the referenced object. The operator writes the corresponding line in the log file: <time stamp> operator* <memory address of referenced entity>

Part 3 Reference counting (2 points)

Add a reference counting mechanism to the Log_Ptr class. The referenced object is only destroyed when count = 0. Implement at least an assignment operator and a copy constructor. Be sure to write the appropriate log messages as well.

Part 4 Thread-safe (1 point)

Make the Log_Ptr class "thread-safe". The reference counting mechanism must work, even if copies of the Log_Ptr-object are used at the same time from different threads.