Shared Mutexes Solutions

Shared Mutex

- Explain briefly what a shared mutex is
  - A shared mutex can be locked in two different ways
  - Exclusive lock. If a thread has an exclusive lock on a shared mutex, no other thread can acquire a lock until the first thread releases the lock
  - Shared lock. If a thread has a shared lock on a shared mutex, other threads can acquire a shared lock without having to wait for the first thread to release it
  - If a thread wishes to acquire an exclusive lock, it must wait until all the threads which have a shared lock release their locks

Single writer, multiple readers

- What is meant by "single writer, multiple readers"?
  - A situation where many threads access shared data, but only a few threads modify it
  - With std::mutex, each thread would have exclusive access to the data, forcing other threads to wait for access
  - It is safe to have multiple threads making interleaved reads (provided there are no modifying threads which could conflict and cause a data race)
  - Giving every thread an exclusive lock causes an unnecessary drop in performance

Single writer, multiple readers

- Explain briefly how this can be implemented without data races using a shared mutex
  - The reading threads acquire a shared lock on the mutex and the writing threads acquire an exclusive lock
  - This allows many threads to read at the same time, when no threads are writing
  - A thread cannot write until all the reading threads have released their locks
  - When a thread is writing, no threads can read until the writing thread releases its lock
  - We can never have a situation in which reading and writing threads conflict

Shared Mutex Example

- Write a program which uses a shared mutex and has two task functions
  - The "writer" task function acquires an exclusive lock on the shared mutex and sleeps for two seconds
  - The "reader" task function acquires a shared lock on the shared mutex and sleeps for two milliseconds
- The program creates 50 "reader" threads, then two "writer" threads, then another 100 "reader" threads
- Check that your program compiles and runs correctly

Shared Mutex Example

- Modify your program to use std::mutex
- Do you notice any difference in performance?
  - This program takes longer to run than the version with a shared mutex
  - This is because the shared mutex allows the reader threads to run concurrently at all times, except when a writer thread has an exclusive lock
  - The std::mutex forces the reader threads to run sequentially all the time. In effect, the program is single-threaded