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 first task function acquires an exclusive lock on the shared mutex and sleeps for two seconds
  - The second task function acquires a shared lock on the shared mutex and does not sleep
- The program creates five threads with the shared lock, then two threads with the exclusive lock, then another five threads with the shared lock
- Add suitable print statements. Explain the results
  - (Note: you may have to force your compiler into C++17 mode)

Shared Mutex Example

- Sample output

```
Read thread 0 with shared lock
Write thread 5 with exclusive lock          // 2 second delay before next output
Read thread 9 with shared lock
Read thread 11 with shared lock
Read thread 1 with shared lock
Read thread 13 with shared lock
Write thread 6 with exclusive lock          //2 second delay before next output
Read thread 7 with shared lock
Read thread Read thread 2 with shared lock
8Read thread 10 with shared lock
 with shared lockRead thread Read thread
Read thread 15 with shared lock
Read thread 4 with shared lock
3 with shared lock
Read thread 14 with shared lock
Read thread 16 with shared lock
12 with shared lock
```

Shared Mutex Example

- Some reader threads run and try to acquire shared locks.
- The first writer thread runs. It waits to acquire an exclusive lock. While it is waiting, no more reader threads can acquire a shared lock.
- When the reader threads release their shared locks, the writer thread acquires an exclusive lock and executes its critical region.
- Reader threads must wait until the writer releases its exclusive lock before they can get a shared lock
- When the writer releases its lock, the reader threads can get shared locks and execute their critical region

std::mutex Example

- Sample output

```
Read thread 0 with shared lock
Read thread 1 with shared lock
Read thread 8 with shared lock
Read thread 12 with shared lock
Read thread 13 with shared lock
Read thread 14 with shared lock
Read thread 7 with shared lock
Read thread 2 with shared lock
Read thread 10 with shared lock
Read thread 11 with shared lock
Read thread 4 with shared lock
Read thread 15 with shared lock
Read thread 16 with shared lock
Write thread 6 with exclusive lock      // 2 second delay before next output
Read thread 9 with shared lock
Read thread 3 with shared lock
Write thread 5 with exclusive lock      // 2 second delay before next output
```

Shared Mutex Example

- Rewrite your program so that it uses std::mutex instead of a shared mutex. Explain the results.
- The output from the reader threads is scrambled up when using shared mutex, but not with std::mutex
  - With a shared mutex, reader threads do not have exclusive access to their critical region and their execution can interleave
  - With std::mutex, reader threads have exclusive access to the critical region. Only one reader thread can execute its critical region at a time
- The output from the writer threads is never scrambled up
  - The writer threads have exclusive access to their critical section in both cases