```
#include <map>
#include <string>
#include <mutex>
#include <shared_mutex>

// Update to code from 2nd edition of Williams' textbook:
// Craig Scratchley, Oct. 2021

class dns_entry
{};

class dns_cache
{
   std::map<std::string,dns_entry> entries;
   std::shared_mutex entry_mutex;
public:
   dns_entry find_entry(std::string const& domain)
   {
      std::shared_lock lk(entry_mutex); // accessing
      auto const it = entries.find(domain);
      return (it==entries.end())?dns_entry():it->second;
   }
   void update_or_add_entry(std::string const& domain,
                            dns_entry const& dns_details)
   {
      std::lock_guard lk(entry_mutex);  // updating
      entries[domain]=dns_details;
   }
};

int main()
{}
```

protecting rarely updated data structures

chap 3.3.2 - p.63

"Shared mutexes are usually used in **situations when multiple readers can access the same resource at the same time without causing data races**, but only one writer can do so." Sep. 5, 2017