```cpp
#ifndef STUDENT_TO_DO_REGION
  /// Include necessary header files
  /// Hint:  Include what you use, use what you include

  #include <cstddef>    // size_t
  #include <filesystem> // exists()
  #include <fstream>
  #include <string>
  #include <utility>    // move()

  #include "BookDatabase.hpp"
#endif



// Return a reference to the one and only instance of the database
BookDatabase & BookDatabase::instance()
{
  std::string filename;
  // Don't forget to #include <filesystem> to get visibility to the exists() function
  //
  // Look for a prioritized list of database files in the current working directory to use
  if     ( std::filesystem::exists( "Open Library Database-Full.dat"   ) )filename = "Open Library Database-Full.dat";
  else if( std::filesystem::exists( "Open Library Database-Large.dat"  ) )filename = "Open Library Database-Large.dat";
  else if( std::filesystem::exists( "Open Library Database-Medium.dat" ) )filename = "Open Library Database-Medium.dat";
  else if( std::filesystem::exists( "Open Library Database-Small.dat"  ) )filename = "Open Library Database-Small.dat";
  else if( std::filesystem::exists( "Sample_Book_Database.dat"         ) )filename = "Sample_Book_Database.dat";


  static BookDatabase theInstance( filename );
  return theInstance;
}



// Construction
BookDatabase::BookDatabase( const std::string & filename )
{
  std::ifstream fin( filename, std::ios::binary );


  #ifndef STUDENT_TO_DO_REGION
    /// The file contains Books separated by whitespace.  A Book has 4 pieces of data delimited with a comma.  (This exactly matches
    /// the previous assignment as to how Books are read)
    ///
    ///       Field           Type           Notes
    ///  1.   Book's ISBN     String         Unique identifier (primary key), always enclosed in double quotes
    ///  2.   Book's Title    String         May contain spaces, always enclosed in double quotes
    ///  3.   Book's Author   String         May contain spaces, always enclosed in double quotes
    ///  4.   Book's Price    Floating Point In dollars
    ///
    ///  Example:
    ///     "0001062417",  "Early aircraft",                "Maurice F. Allward", 65.65
    ///     "0000255406",  "Shadow maker \"1st edition)\"",  "Rosemary Sullivan",   8.08
    ///     "0000385264",  "Der Karawanenkardinal",          "Heinz Gstrein",     35.18
    ///
    ///  Note: double quotes within the string are escaped with the backslash character
    ///
    Book book;
```

```
62      while( fin >> book )
63      {
64        // All components of the compound data type Book found in the input stream and read without error, so move the complete
65        // book into the memory resident data store.  Incomplete Books have already been rejected.
66        _data.emplace_back( std::move(book) );  // or   _data.push_back( book );
67      }
68    #endif
69
70    // Note:  The file is intentionally not explicitly closed.  The file is closed when fin goes out of scope - for whatever
71    //        reason.  More precisely, the object named "fin" is destroyed when it goes out of scope and the file is closed in the
72    //        destructor. See RAII
73  }
74
75
76  #ifndef STUDENT_TO_DO_REGION
77    /// Implement the rest of the interface, including functions find (recursively) and size
78    ///
79    /// Programming note:  An O(n) operation, like searching an unsorted vector, would not generally be implemented recursively.  The
80    ///                    depth of recursion may be greater than the program's function call stack size.  But for this programming
81    ///                    exercise, getting familiar with recursion is a goal.
82
83    Book * BookDatabase::find( const std::string & isbn )
84    {
85      // Delegate to the recursive function providing a starting point
86      return find( isbn, _data.begin() );
87    }
88
89
90    Book * BookDatabase::find( const std::string & isbn, std::vector<Book>::iterator currentBook  )
91    {
92
93      if( currentBook          == _data.end() ) return nullptr;           // Base case
94      if( currentBook->isbn() == isbn         ) return & *currentBook;    // visit the node (return the address of the current book)
95      return find( isbn, std::next( currentBook ) );                      // recurse with a smaller, simpler problem to solve
96    }
97
98
99    std::size_t BookDatabase::size() const
100   {
101       return _data.size();
102   }
103 #endif
104
```