```cpp
#include <cstddef>      // size_t
#include <fstream>
#include <string>
#include <utility>      // move()

#include "GroceryItemDatabase.hpp"





// Return a reference to the one and only instance of the database
GroceryItemDatabase & GroceryItemDatabase::instance( const std::string & filename )
{
  static GroceryItemDatabase theInstance( filename );
  return theInstance;
}





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


  #ifndef STUDENT_TO_DO_REGION
    /// The file contains one record of data on each line of text.  See Grocery_UPC_Database_Sample.dat.  A record has 4 pieces of
    /// data delimited with a comma.  (This exactly matches how Grocery Items are read)
    ///
    ///       Field            Type            Notes
    ///  1.   UPC Code         String          Unique identifier (primary key), always enclosed in double quotes
    ///  2.   Brand Name       String          May contain spaces, always enclosed in double quotes
    ///  3.   Product Name     String          May contain spaces, always enclosed in double quotes
    ///  4.   Price            Floating Point  In dollars
    ///
    ///  Example:
    ///    "00024600017008",   "Morton",         "Morton Kosher Salt Coarse",                                    15.17
    ///    "00033674100066",   "Nature's Way",   "Nature's Way Forskohlii - 60 Ct",                              6.11
    ///    "00041520893307",   "Smart Living",   "Smart Living 10.5\" X 8\" 3 Subject Notebook College Ruled",   18.98
    ///
    ///  Note: double quotes within the string are escaped with the backslash character
    ///
    GroceryItem item;

    while( fin >> item )
    {
      // All components of the compound data type Grocery Item found in the input stream and read without error, so move the
      // complete grocery item into the memory resident data store.  Incomplete Grocery Items have been rejected.
      _data[ item.upcCode() ] = std::move( item );   // or   _data[ item.upcCode() ] = item;
    }
  #endif

  // Note:  The file is intentionally not explicitly closed.  The file is closed when fin goes out of scope - for whatever
  //        reason.  More precisely, the object named "fin" is destroyed when it goes out of scope and the file is closed in the
  //        destructor. See RAII
}





GroceryItem * GroceryItemDatabase::find( const std::string & upc )
{
  #ifndef STUDENT_TO_DO_REGION
    /// Search the memory resident container named "_data" looking for a grocery item with a matching UPC.  If found, return a pointer
    /// to that grocery item.  Otherwise return a null pointer.
    ///
    /// Hint:  Don't walk the list (O(n) operation), find the item with a binary search (O(log n) operation)
    auto item = _data.find( upc );

    if( item == _data.end() ) return nullptr;
    else                      return & (item->second);
  #endif
}





std::size_t GroceryItemDatabase::size() const
{
  // Returns the number of grocery items in the grocery item database.
  #ifndef STUDENT_TO_DO_REGION
    /// Delegate the actual work of determining how many items are stored to the underlying container named "data"
    return _data.size();
  #endif
}

```