```cpp
//
//book.h
//
#ifndef BOOK_H
#define BOOK_H

#include <iostream>
#include <string>

class Book
{
private:
    std::string title;
    std::string last;
    std::string first;

public:
    Book(const std::string& ttl, const std::string& lst, const ↩
    ↪ std::string& fst)
        : title(ttl), last(lst), first(fst) {}

    std::string get_title() const
    {
        return title;
    }
    std::string get_last() const
    {
        return last;
    }
    std::string get_first() const
    {
        return first;
    }
};

inline std::ostream& operator<<(std::ostream& ostr, const Book& ↩
↪ book)
{
    ostr << "{TITLE: '" << book.get_title()
         << "', LAST: '" << book.get_last()
         << "', FIRST: '" << book.get_first() << "'}";
    return ostr;
}
#endif // BOOK_H
```

```
//
//catalogue.h
//
#ifndef CATALOGUE_H
#define CATALOGUE_H

#include <string>
#include <vector>
#include "Book.h"

class Catalogue
{
private:
    std::vector<Book *> booklist;
    static bool equal_ignore_case(const std::string& string1, ↩
    ↪ const std::string& string2);

public:
    void add(const std::string& title, const std::string& last, ↩
    ↪ const std::string& first);
    std::vector<Book *> find(const Book& target) const;
};
#endif // CATALOGUE_H

//
//catalogue.cpp
//
#include <string>
#include "Book.h"
#include "Catalogue.h"
using namespace std;

bool Catalogue::equal_ignore_case(const string& string1, const ↩
↪ string& string2)
{
    if(string1.size() > string2.size()) return false;

    for (int i = 0; i < (int)string1.length(); i++)
    {
        if (tolower(string1[i]) != tolower(string2[i]))
        {
            return false;
        }
    }
```

```
    return true;
}

void Catalogue::add(const string& title, const string& last, const string& first)
{
    booklist.push_back(new Book(title, last, first));
}

vector<Book *>Catalogue::find(const Book& target) const
{
    vector<Book *> matches;
    string target_title = target.get_title();
    string target_last = target.get_last();
    string target_first = target.get_first();
    for (Book *book : booklist)
    {
        if ( equal_ignore_case(target_title, book->get_title())
                && equal_ignore_case(target_last, book->get_last())
                && equal_ignore_case(target_first, book->get_first()))
        {
            matches.push_back(book);
        }
    }
    return matches;
}
//
//main.cpp
//
#include <iostream>
#include <vector>
#include <string>
#include "Book.h"
#include "Catalogue.h"
using namespace std;

void fill(Catalogue& catalogue)
{
    catalogue.add("Life of Pi", "Martel", "Yann");
    catalogue.add("The Call of the Wild", "London", "Jack");
    catalogue.add("To Kill a Mockingbird", "Lee", "Harper");
    catalogue.add("Little Women", "Alcott", "Louisa");
```

```
    catalogue.add("The Adventures of Sherlock Holmes", "Doyle", ↩
    ↪ "Conan");
    catalogue.add("And Then There Were None", "Christie", "Agatha");
    catalogue.add("Carrie", "King", "Stephen");
    catalogue.add("It: A Novel", "King", "Stephen");
    catalogue.add("Frankenstein", "Shelley", "Mary");
    catalogue.add("2001: A Space Odyssey", "Clarke", "Arthur");
    catalogue.add("Ender's Game", "Card", "Orson");
}

void search(const Catalogue& catalogue, const Book& target)
{
    cout << endl << "Find " << target << endl;
    vector<Book *> matches = catalogue.find(target);
    if (matches.size() == 0) cout << "No matches." << endl;
    else
    {
        cout << "Matches:" << endl;
        for (Book *book : matches)
        {
            cout << " " << *book << endl;
        }
    }
}

void test(const Catalogue& catalogue)
{
    Book target1("Life of Pi", "Martel", "Yann");
    search(catalogue, target1);
    Book target2("", "King", "");
    search(catalogue, target2);
    Book target3("1984", "Orwell", "George");
    search(catalogue, target3);
    Book target4("", "", "");
    search(catalogue, target4);
}

int main()
{
    Catalogue catalogue;
    fill(catalogue);
    test(catalogue);
}
```