CS 200: Introduction to Programming

Dr. Naveed Arshad

Assignment 6

(Due Date: Thursday, December 17, 2015)

This assignment is due on Thursday, 17 December. The usual late submission policy of 10% per day deduction for up to 3 days applies. The assignment needs to be submitted on LMS under the ‘Assignments’ tab.

The course policy about plagiarism is as follows:

1. Students must not share actual program code with other students.
2. Students must be prepared to explain any program code they submit.
3. Students cannot copy code from the Internet.
4. Students must indicate with their submission any assistance they received.
5. All submissions are subject to automated plagiarism detection.

Students are strongly advised that any act of plagiarism will be reported to the Disciplinary Committee.

Templates

Task-1 [50 points]

For this brief assignment, you are to implement a generic ordered set abstraction. As a starting point, we are providing code for a set of integers. The provided code implements the following operations:

```
    oset();          // default constructor -- empty set
    oset(int);       // constructor for singleton set
    oset(oset&);     // copy constructor

    bool operator[](const int)      // find:    if (S[3]) ...
    oset& operator+=(const int)     // insert:  S += 3
    oset& operator-=(const int)     // remove:  S -= 3

    oset& operator+=(oset&)         // union
    oset& operator-=(oset&)         // set difference
    oset& operator*=(oset&)         // intersection
```

The provided code also provides a simple iterator mechanism (still part of class oset):

```
    class oset::iter;               // type declaration
    iter begin()                    // returns "pointer" to
first element of set
    iter end()                      // returns pointer beyond
end of set
```

Iterators themselves support the following methods:

```
    const int& operator*()          // "dereference" the
iterator
    iter& operator++()              // prefix ++; point to next
element
    iter operator++(int)            // postfix ++; point to next
element
        // NB: the int arg is unused; C++ uses it by convention
to tell
        // the difference between prefix and postfix increment
    bool operator==(iter)             // do iterators point
    bool operator!=(iter)             //    at the same element?
```

Note that the `iter` constructor in inaccessible to user code: the only way to get an iterator is by calling `begin()` or `end()`. For what it's worth, `oset::iter` resembles what the C++ standard template library calls a "forward, constant" iterator: it can be incremented but not decremented, and the "pointed-at" elements are read-only.

This task has two parts:

1. The set code should be generic. Rewrite it as a template in which the existing `int` types are replaced with a type parameter `T`. Test your code on `doubles` and `string`.

2. The private `find_prev` method, called by `operator[]()`, `operator+=(T)`, and `operator-=(T)`, where `T` is your element type, implicitly assumes the existence of a >= operator for`T`. Rewrite it to make ordering of `T` explicit by passing a ccomparator function to (each of) the `oset` constructors. Your comparator should return −1, 0, or 1 depending on whether its first argument is less than, equal to, or greater than its second. Test your code on strings with both case-sensitive and case-*in*sensitive lexicographic ordering.

Special Notes

For purposes of this task, you are not permitted to use any of the collection classes in the C++ standard library (or in any third-party library) to implement set functionality. (Using them for testing or for the creation of element types is ok.)

Task-2 [50 points]

Your friend Bob is developing an inventory management system for a bookstore. The system provides the following features: adding items to the inventory, removing items from the inventory, listing items in the inventory, and looking-up items in the inventory by their ISBN number or by keyword.

Bob already developed most of the system, but his software has some bugs and some features are still missing. Bob does not understand what is wrong with his system and he comes to you for help. In this assignment, you will help Bob debug his system and complete its implementation.

Before working on this assignment, please read the following short tutorials on using the STL map class and the STL list class. Note that there is a typo in the list tutorial. In the last example, on the line that prints each element in the list, they forgot to dereference the iterator.

In the code that Bob wrote, several member functions are declared as "const" (by appending the keyword "const" to the end of their prototype). These member functions are simply not allowed to modify any data members in the class. Some arguments are also declared as "const". These arguments cannot be changed by the functions that receive them. Using such restrictions is a good programming practice. We will soon talk about it in class.

1. Exploring the code

Examine the source code provided to you by Bob.

Using the provided main.cpp, build the bookstore program, execute the program, and try to list the current inventory (do not try other options yet, they have bugs in them). You should see three items in the inventory.

If you examine main.cpp, you will notice that two books and one movie are inserted into the inventory. When you list the inventory, the books look fine, but the movie does not display itself properly. Only the generic attributes of an Item object are displayed.

Modify the class Movie to ensure that printing the content of the inventory displays movies with all their attributes. Use the Book class as an example.

2. Passing objects by value and by reference

Execute the bookstore program again. This time, try not only to list the inventory but also try to add a few books or movies.

As you can notice, the addBook and addMovie functions inside the file main.cpp seem to have bugs in them. For example, addBook prompts the user for information about the new book to add to the inventory, but it fails to actually update the inventory. Help Bob by doing the following:

- Fix the addBook function in the file main.cpp. Functions addMovie and removeItem have the same bug in them. Fix them as well.
- addMovie seems not to have been implemented at all. Bob says that he is too tired to implement it. Help Bob and implement this function for him.

3. Using the Standard Template Library

Map class

Bob did not yet implement the Inventory::removeItem function. Bob does not know how to do it. Help Bob and implement this function for him. The function should print an error message if the item is not in the inventory. Otherwise, the function should remove the item from the_table and **should delete the Item object**.

Hints:

- Examine the other member functions in class Inventory to see how they manipulate the _table map. In particular, examine the function lookupItem.
- Rember the short tutorials on using the STL map class.

List class

In the function lookupItemsByKeyword in the file main.cpp, Bob forgot to print the results returned by the inventory. Add the missing functionality to this function.

Hint:

- Remember the short tutorials on using the STL list class. Note that there is a typo in this tutorial. In the last example, on the line that prints each element in the list, they forgot to dereference the iterator.