CS 304
Program 3: Templates and Iterators

Due Wednesday, Oct. 9, 10:00 pm

This project is worth a total of 80 points, or 8% of your course grade

Goals

- Implement and test a container class in C++ that uses templates and iterators.

Reading

Main & Savitch:
- Ch. 3.4
- Ch. 6
- Appendix H (especially page 803)

Overview

Convert your bag class from assignment 2 (bag_with_receipts) to use templates, and add bi-directional const and non-const iterators.

Details

Use your completed Assignment 2 (bag with receipts) as a starter file.
Note: If you did not complete this assignment, see me for a working version.

Make bag a template class (25 points total)

Rename bag2.h and bag2.cpp to bag.h and bag.cpp. Leave the namespace as cs304_bag.

Convert your bag container class to a template class (see summary on page 304). Don't forget to update the API.

In Visual Studio, you will need to change the type of the bag.template file from "C/C++ compiler" to "C/C++ header." You can do this as follows:
- Select the bag.cpp file.
- Right-click to bring up the contextual menu.
- Select Properties.
- Use the pull-down menu to change Item Type from "C/C++ compiler" to "C/C++ header."
- Click Apply.
- Rename the file to bag.template.
- In the Solutions Explorer pane, move bag.template from Source Files to Header Files.

Make sure your bag class still works correctly before starting the next part of this assignment.

Add bi-directional iterators (45 points total)

Add a bi-directional iterator to your bag class. This iterator should allow users of your bag class to retrieve each Item from the bag, skipping over not-in-use slots in the bag array. The class definition can go into bag.h and bag.template (if not inlined).

Your iterator class should meet the following conditions (20 points total)

a. Name the iterator bag_iterator. bag_iterator should be a child class of std::iterator:

```
template <class Item>
class bag_iterator
: public std::iterator<bi_directional_iterator_tag, Item>
```

b. The iterator's state will consist of pointers to the two arrays that represent the bag's state, plus a pointer to the bag's capacity. The iterator will also need to keep track of the current item. (2 point2)

c. All iterators must provide the following functions:

- A constructor with three parameters - pointers to the bag's arrays and capacity - that defaults to an invalid ("end') iterator value. (2 points)
- operator == and != return true if two iterators point to the same Item. (3 points)
- operator * returns the Item that the iterator currently points to (3 points)

d. Your iterator is a bi-directional iterator, so it also must provide prefix and postfix ++ and -- operators (10 points total). These will return a pointer to the next valid bag element in the appropriate direction, skipping over not-in-use positions, or return an invalid iterator (hint: use the default constructor) if there are no more valid Items in that direction.

In the bag class, add (10 points total)

e. A typedef to protect users of your bag class from having to know the exact type of the iterator. (3 points)

f. An end() function that returns an invalid iterator (hint: use the default constructor). (2 points)

g. A begin() function that returns an iterator to the first valid Item in the bag, or returns an invalid iterator. (5 points)

TEST your iterator thoroughly before continuing. Really.

h. Repeat the steps above to create a const_bag_iterator.
(15 points total).
Note: This is largely cut-and-paste, so be sure that you have thoroughly tested and debugged your non-const iterator before doing this step.

Test the const iterator to satisfy yourself that it works correctly.

Quality (10 points total)

- Code conforms to (updated-for-assignment 2) C++ coding standards (10 points)

Submitting

Zip up the following files ONLY: bag.h, bag.template, stdafx.h, your test program. Submit the zipped file through Moodle.