CISC 2200—Data Structures
Spring, 2016

Programming Project # 2: Extending the Bag ADT
(Based on Exercises 1.6–1.8, along with Programming Problem 3.5)
Date Due: Wednesday 2 March 2016

In CISC 1100 or 1400, you learned how to compute the union, intersection, and difference of sets:

$$A \cup B = \{\, x : x \in A \wedge x \in B \,\}$$
$$A \cap B = \{\, x : x \in A \vee x \in B \,\}$$
$$A - B = \{\, x : x \in A \wedge x \notin B \,\}$$

The union, intersection, and difference of bags are defined analogously. The only real difference is that since bags may contain repeated elements, the union (or intersection or difference) of bags may contain repeated elements. For example, suppose that we define bags

$$A = [1, 1, 2, 2, 3, 4, 4, 4] \qquad \text{and} \qquad B = [1, 2, 3, 3, 4, 4, 5].$$

(Here, we use $[\ldots]$ to denote bags, similarly to the way that $\{\ldots\}$ denotes a set. Also, there's no reason to assume that a bag is sorted; we show the bag elements in sorted order simply to make it easier to follow the example.) Then

$$A \cup B = [1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5]$$
$$A \cap B = [1, 2, 3, 4, 4]$$
$$A - B = [1, 2, 4]$$

See Exercises 1.6–1.8 for further discussion. The purpose of this project is to add these operations to our Bag ADT.

Before doing so, note that the author has given us two different implementations: array-based and link-based. The main problem with the former is that the implementor must decide how big to make the underlying array. This problem doesn't exist for the latter; however, linked structures are more complicated than arrays. However, if we were to use a vector-based implementation, then we wouldn't need to worry about how to choose the size of an array; moreover, vectors are easier to handle than linked structures. So the first thing to do is to create `VectorBag`, a vector-based implementation. Since vectors may be thought of as "safe arrays", the best thing to do would be to start with the array-based implementation and work from there.

Once you've done this, you should now add new operations to the `VectorBag` class. Since the standard symbols $\cup$ and $\cap$ are not available on our keyboard and since `union` is a reserved word, let's use `+` and `*` to denote union and intersection;[1] of course, `-` is the obvious choice for denoting set difference. This means that you'll want to define `operator+` (and so forth), making sure to add appropriate `doxygen` comments that describe what these do.

Your main task will be to design, write, and test the `VectorBag` class. Your solution will contain two files:

- `VectorBag.h`: the interface file for the `VectorBag` class.

[1] This notation is actually used in some books.

- `VectorBag.cpp`: the implementation file for the `VectorBag` class.

To aid you in same, I am providing you with the following files, which may be found in the "share directory" `~agw/class/datastr/proj2` for this project:

- A `Makefile` for this project.
- The file `proj2.cpp`, for testing your `VectorBag` class. This file is similar to the `main.cpp` that the authors provide for testing the `ArrayBag` class. Of course, it works for `VectorBag`, rather than for the `ArrayBag` class. In addition:
  - The `displayBag()` function will work for a `VectorBag<ItemType>`. It also displays the items in sorted order. Hence the < operation must be defined for `ItemType`.
  - The new operations ($\cup$, $\cap$, $-$) are tested.
- A working version of the program, called `proj2-agw`, for you to try out.
- Your recollection of overloaded operators (such as `operator+` and suchlike) may be a bit rusty. I have provided you with a file `point.cc`, which defines a `Point` class that has an `operator+` (for adding two `Points`).

With the exception of `point.cc`, all these should be copied to your working directory for Project 2.

You may consider your program correct when it compiles with no warnings, and it produces the same output as `proj2-agw`.

Deliverables: You are to do the following:

1. Use `doxygen` comments to document your program, and use `doxygen` to create an `html` documentation tree. For the latter, I'd suggest copying your Project 1 `Doxyfile` over to your Project 2 working directory, and changing the one appearence of "`proj1`" in the `Doxyfile` to "`proj2`". Once again, you may see my `doxygen`-generated documentation, which resides at

   `http://www.dsm.fordham.edu/~agw/datastr/doxygen/proj2`,

   to see how this should look.

2. Submit a clean typescript, containing the following commands (and their output):

   ```
   cat VectorBag.h
   cat VectorBag.cpp
   make clean
   make
   proj2
   ```

   I prefer an email submission. Please make sure that the subject of your email msg is "`Project 2`" and that your return address is correct. (See the Departmental help document for details.)

Remark: All three operations can be done using the original operations named in `BagInterface.h`. (As a matter of fact, this is probably the simplest way to implement these operations.) One side benefit: If you wanted to add these operations to the `LinkedBag` class, the same code should work, since your code would be independent of the underlying implementation.

Good luck!!