Name:		
(as it would appear on official course roster)		
Umail address:	@umail.ucsb.edu	section 9am, 10am, 11am, 12pm
Optional: name you wish to be called if different from name above.		
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"		

1

h14

CS32 F19

h14: Container Classes Review

ready?	assigned	due	points
true	Tue 11/12 02:00PM	Tue 11/19 02:00PM	50

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE LISTED ABOVE AS THE DUE DATE,
OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments;
in place of that, we drop the lowest scores (if you have zeros, those are the lowest scores.)

Reading: Container Classes Review, DS Chapter 3, and the assert() function on p. 793

1. (10 pts) Fill in the information in the header. The following are required to get the 10 "participation" points.
    - Filling in your name and umail address.

   Also: For paper submission PLEASE submit on ONE SHEET OF PAPER, double-sided if at all possible. If you must submit on two printed sheets write name on BOTH sheets and no staples, paperclips, or folded corners.

   Chapter 3 is a chapter you likely already read during CMPSC 24, but we want to review the material now. You will likely have a deeper appreciation and understanding of the material now, from the perspective of greater familiarity with C++ classes, objects, algorithm analysis, and data structures. The questions below emphasize a few important points we are building on and reinforcing in CMPSC 32.

   In particular, I want to make sure you can demonstrate that you understand:
    - the concept of a `typedef`, and the way it lays the foundation here for template classes.
    - the `std::size_t` data type and why it is used.
    - the idea of *value semantics* as used by the authors of DS (which is important to understanding copy constructor and overloaded = operator as discussed in DS).
    - how operator overloading works (the example of the `operator +` used in this chapter.
    - the way the `static` member constant is used to define a capacity.
    - the notion of a "partially filled array", i.e. occupancy vs. capacity
    - the circumstances in which a value type must have a default constructor (p. 110).
    - the way the `assert` macro is used in this code example.
    - short circuit evaluation as a guard against invalid operations (pp. 114-115).
    - the use of the STL multiset template class

   There are MANY more things in this chapter that are also important. Space and time prohibit me from asking a specific homework question about each of them. I encourage to read the WHOLE chapter carefully rather than just skimming for the answers to the questions on the back, and be sure you understand *at least* all of the items listed above. I may ask you about *anything* in this chapter on the midterm exam (well, this should all be review by now), even if is not specifically covered on this homework assignment, but *especially* the items in the list above.

   After reviewing the chapter, answer the questions on the next page:

2. (6 pts) In the example `Bag` class in this chapter, the authors use the statement:

```
typedef int value_type;
```

What do the authors cite as the advantages of doing this typedef over just using `int` throughout the definition of the `Bag` class?

2

h14

CS32 F19

3. (6 pts) The `size_t` type is frequently defined in C/C++ software. What advantage does it have over just using `int`? (Note: the textbook offers an explanation, and we'll accept the textbook's explanation as a sufficient answer, but if you want to really understand, you may need to do some research online on your own.)

4. (6 pts) Another potential use of a typedef is to shorten a type. For example, we can have a function with this prototype that uses `std::pair<double,double>` to represent an (x,y) point in the cartesian plane:

```
double distanceBetween(pair<double,double> p1, pair <double,double> p2);
```

Write a `typedef` statement that would allow us to write this as:

```
double distanceBetween(Point p1, Point p2);
```

5. The authors of DS frequently talk about the *value semantics* of a data type. At the top of figure 3.1, they indicate that the `typedef` for `value_type` may be any of the C++ built-in types (int, char, etc.) **OR** any type that meets four criteria. What are the four criteria that indicate what the type must meet? List them below.

    a. (3 pt)

    b. (3 pt)

    c. (3 pt)

    d. (3 pt)

    e. (5 pts) The textbook discussion goes on to explain that the assignment operator is NOT defined for every data type, and they cite an example. What example do they cite?

6. (5 pts) What is the purpose of the `assert()` macro?