Dynamic Memory And Test Drivers, Reading

Now that we've added three new functions, yes -- they do have to be included in test drivers.

Testing The Destructor Function

This is the easy part -- there's no code to add! If your test driver runs and terminates with no errors, then either (1) your destructor function was successfully called or (2) you forgot to write a destructor function! If you're thinking there's really no good way to see if the destructor worked or not, you're right. All you can do is look at its coding. But the good news is that its coding is really short and not very complicated at all. And if you did anything seriously wrong, your program will crash.

Testing The Copy Constructor Function: The "Object Copy Test"

A statement like `Array<int> d = a;` or `Array<int> d(a);` invokes the copy constructor, if there is one -- otherwise it just copies the data members. The test is done with assertions -- no actual and expected output:

```
// object copy test
cout << "\nObject copy test\n";
Array<int> d = a; // making a copy
assert(a.capacity() == d.capacity());
for (int i = 0; i < a.capacity(); i++)
  assert(a[i] == d[i]); // uses the setter version for both a and d
```

Testing The Assignment Operator Function: The "Object Assignment Test"

This looks like it's the exact same as the object copy test, but it's not. A stand-alone *re*assignment of an object uses the operator= function instead of the copy constructor:

```
// object assignment test
cout << "\nObject assignment test\n";
Array<int> e; e = a;
assert(a.capacity() == e.capacity());
for (int i = 0; i < a.capacity(); i++)
  assert(a[i] == e[i]);
```

It's *very important* that this be `Array<int> e; e = a;`, and not `Array<int> e = a;` because those are two entirely separate ways to make a copy of an object, and they involve different functions.

Complete List Of Tests -- An Updated Checklist

Here are the tests you'll do in any test driver:

1. Test all public functions to make sure they work as you expect them to work.
2. Const object test. (like `const Array<int> c;`)
3. **Object copy test. (like `Array<int> d = a;`)** *new*
4. **Object assignment test. (like `Array<int> e; e = a;`)** *new*
5. The "ifndef" test.