C SC 397a, Spring 2010
Notes on Assignment 6 Grading

Problem 1. (Books)

The tests made available in $FILES/a6 before the due date accounted for 83 of this problem's 100 points. I added three more bx-based test files that contained randomly generated data. Each was worth one point. See a6/bxg.? and, if you're curious, a6/bxgen.icn, the Icon program that generated the data.

Memory management

There are lots of elaborate tools for analyzing memory leaks and other memory management errors but I took a low-tech approach for investigating memory management by asking a simple question: Does a trivial computation run repeatedly reach steady state memory utilization? Here's the program I used to pose that question:

```
$ cat $FILES/a6/bookloop.cc
#include <cstdio>
#include "Books.h"

int main()
{
   while (1) {
      Book bk("a", 2, 1);
      Bookcase b(1);
      for (int i = 1; i <= 10; i++)
         b.addShelf(100, 10*i);

      for (int i = 1; i <= 1000; i++)
         b.add(&bk);
      }
}
```

The command "ps u" was used to observe whether memory utilization for a run of bookloop reached a steady state. Of 21 solutions, only six reached steady state. Four terminated with a fault of some sort. The remaining eleven steadily consumed memory.

This bookloop test was worth 14 points, all or nothing. The result of this testing is reported under BooksMMTest. I encourage you to try the bookloop.cc test yourself and observe with "ps u", regardless of whether you passed it.

You should expect a similar, simple minded test to be part of the suite used for grading String and IntList on assignment 7.

As the syllabus states, you're free to exchange test cases with classmates as long as the tests aren't part of the deliverables for the assignment. Feel free to limit distribution to your friends, or make new friends by posting tests to the mailing list. I'll feel free to include any tests posted to the mailing list as part of the assignment 7 test suite used for grading.

Data Representation

We'll soon see that templated container classes are the right way to represent collections in C++ but I haven't made those available to you yet. In lieu of that soon-coming technology, nine of the 21 solutions used a linked list of some sort to handle the requirement that a shelf can hold an unlimited number of books; the balance used an expanding-array approach, similar to the implementation of Java's Vector class. If you studied my solution you saw that's what I did, too.

A few students mentioned that it's a little clumsy to implement a linked list without using any public data members. One way to work around that is to use friends. Another approach is to have member functions return references to data members that are pointers. The merit of that is debatable, since returning a reference to a data member is not far from just making it public. We haven't talked about references to pointers but here's an example of using one—the ptr() member function below returns a reference to itsPtr:

```
#include <cstdio>

class X {
  public:
    char* & ptr() { return itsPtr; }
  private:
    char *itsPtr;
  };

int main()
{
  X x;
  x.ptr() = "abc";

  X y;
  y.ptr() = "xyz";

  puts(x.ptr());
  puts(y.ptr());
}
```

Dealing with big diffs

There were some complaints about having to deal with big diffs. The fact is that the computing world is full of big diffs that are getting bigger every day and it's important to develop the skills to deal with them. Along with various approaches using multiple editor views, or Eclipse, or visual diff'ing tools, here's an example of a simple shell-based approach that works well in many cases:

```
$ diff <($FILES/a6/b2 | head -20) <(b2 | head -20)
```

The above command runs that pesky b2, which produces 100,003 lines of output (and perhaps a 200,000 line diff), and uses process substitution to diff only the first 20 lines. You could then use a history substitution like !!:gs/20/500/ to diff the first 500 lines instead, or use a shell variable,

```
$ N=500 diff <($FILES/a6/b2 | head -$N) <(b2 | head -$N)
```

or just wrap it in a trivial script that passes a command line argument along to the invocations of head.

Odds and ends

I'm sorry to say that a few students got the impression that *"You may not use any C++ library classes such as string, vector, etc."* meant that strcpy(), strlen(), et al. couldn't be used and wrote out loops instead. Those aren't classes; they're functions, and were fine to use.

Mr. Wheelwright reminded me of a g++ misbehavior I'd seen before but forgotten about: By default g++ doesn't warn about a missing return at the end of a function:

```
$ cat nowarning.cc
bool f() {}
int f2() {}
char* f3() {}
$ g++ -c nowarning.cc
$                          (Silence is not golden!)
```

Using g++ -Wall will make it complain in this case, and many others.

Miscellaneous

As usual, there were some number of extra.txt comments of the form "I was unable to complete <some problem>." Remember that assignments aren't take-home tests—my goal is that everybody get 100% on every problem of every assignment. If you're stuck, or frustrated, and especially if you're about to write off a problem, let me know, no matter how close to the deadline it is, or how late a start you got, or whatever. Give me a chance to help you get 100% the rest of the way out!

There were 16 extra.txt submissions that had a clear estimate of hours spent on the whole assignment and that didn't cite factors that could add significant noise to the data. Here are the data points: 3, 3, 3, 4, 4, 4.5, 4.5, 5, 5, 5, 5.5, 6, 9, 10, 12, 18. The median is 5 hours; the mean is 6.3 hours.