C SC 397a, Spring 2010
Assignment 6
Due: Monday, March 22 at 22:00:00

Problem 1. (100 points) Books

In this problem you are to use C++ to implement a model of a slightly intelligent bookcase. A bookcase has up to ten shelves. Books have thickness and weight; shelves have width and capacity (a maximum weight of books). A bookcase has the responsibility of determining on which shelf a new book is placed. Adding a book to a shelf decreases the available space and capacity.

You are to implement two C++ classes, Book and Bookcase, with the interfaces described below. You may introduce other classes, a Shelf class, for example, if you wish. The test programs will exercise your code by interacting with only the Book and Bookcase classes.

Here are the interfaces for Book and Bookcase:

```
class Book {
   public:
      //
      // Create a book with the given name, thickness (in millimeters), and
      // weight (in dekagrams). Book names are not necessarily unique.
      //
      // Note: Book holds a copy of name.
      //
      Book(const char *name, int thickness, int weight);
      ~Book();

      //
      // Accessors
      //
      const char *name();
      int thickness();
      int weight();

      //
      // Prints information about the book using this format:
      //   "%s (%d mm, %d dg)\n"
      //
      void print();
      };
```

```
class Bookcase {
  public:
    //
    // Create a Bookcase with the given numeric id
    //
    Bookcase(int id);
    ~Bookcase();

    //
    // Add a shelf with the given width (in millimeters) and capacity (in
    //  dekagrams).
    //
    // A bookcase initially has no shelves, and a maximum of ten shelves.
    //
    // A shelf may be added to a Bookcase that already has some books.  The
    // books are not rearranged but subsequent book additions take the new
    // shelf into account.
    //
    // addShelf returns false iff the addition would create an eleventh shelf.
    //
    bool addShelf(int width, int capacity);

    //
    // Add the book referenced by bp to the bookcase.
    //
    // The book is added to the shelf that has the most available space, but only
    // shelves that have the capacity to accommodate the additional weight of the
    // book are considered.  If two or more shelves have the same amount of available
    // space, the book is added to the oldest (i.e., first added) suitable shelf.
    //
    // A book may be in more than one bookcase.
    //
    // Note: Destroying the bookcase does not destroy the books.
    //
    //  add returns true if the Book was added successfully.  It returns false if
    //  there is no shelf capable of holding the book.
    //
    bool add(Book *bp);

    //
    // Print the contents of the bookcase shelf by shelf
    //
    // The first line of output has the form "Bookcase #N".
    //
    // The contents of each shelf are preceded with a line with this format:
    //    "--- Shelf (%d mm, %d dg) ---\n",
    // where the values are the width and capacity of the shelf.
    //
    // Each book is then printed by simply calling its print() method, but the
    // print() output is preceded by the ordinal position on the shelf, printed with
    // this format: "%2d: "
    void print();
```

```
        //
        // Return the id specified in the constructor.
        //
        int id();
};
```

A simple example: (yes, it has some memory leaks!)

```
// - - - b1.cc - - -
#include <cstdio>
#include "books.h"
int main()
{
   Bookcase b(1);
   b.addShelf(300, 100); // 300mm wide, can hold 100 dekagrams
   b.addShelf(200, 200);

   b.add(new Book("a", 20, 50)); // 20 mm wide, weighs 50 dg
   b.add(new Book("b", 30, 80)); // too heavy for 1st shelf--exceeds capacity
   b.add(new Book("c", 40, 40)); // first shelf has most space and
                                 // adequate capacity

   b.add(new Book(".NET in a Nutshell", 100, 400)); // too heavy--ignored
   b.print();

   puts("\nSupershelf to the rescue!");

   b.addShelf(150, 1000); // Add Supershelf!
   b.add(new Book(".NET in a Nutshell", 100, 400));

   b.print();
}
```

Output:

```
Bookcase #1
--- Shelf (300 mm, 100 dg) ---
 1: a (20 mm, 50 dg)
 2: c (40 mm, 40 dg)
--- Shelf (200 mm, 200 dg) ---
 1: b (30 mm, 80 dg)

Supershelf to the rescue!
Bookcase #1
--- Shelf (300 mm, 100 dg) ---
 1: a (20 mm, 50 dg)
 2: c (40 mm, 40 dg)
--- Shelf (200 mm, 200 dg) ---
 1: b (30 mm, 80 dg)
--- Shelf (150 mm, 1000 dg) ---
 1: .NET in a Nutshell (100 mm, 400 dg)
```

Please make careful note of the following:

The number of books a shelf can hold should be limited only by available memory.

A bookcase might be directed to add a book that has already been added to it. If so, simply treat it as another book. (That is, don't check for duplicates, simply hold the Book pointers that are supplied.)

You may not use any C++ library classes such as string, vector, etc. **Using a C++ library class may result in a score of zero for this problem.**

All source code must be contained in two files: Books.h and Books.cc. You may distribute code between the two files as you see fit.

Book and Bookcase may have no public member functions other than those specified above. No data members can be public.

Be sure to avoid memory leaks.

You must make good use of C++, including but not limited to:

- Inlining member functions when appropriate
- Using bool instead of integers when appropriate
- Using member initializers when appropriate
- No public data members
- Using new and delete rather than malloc and free

Test programs of the form b*.cc and corresponding executable reference versions are in $FILES/a6. Note that bx.cc reads standard input. The bx.N files are input for bx and are used like this:

```
$ bx < bx.1
...output...
$
```

Each of the b*.cc and bx.N files have a comment designating the point value. Each is all or nothing for the specified point value; if there's a diff, even one byte, it'll be zero points for that one. The total of the supplied tests will be less than the full value of this problem. The remaining points will be determined by code quality and some additional tests.

$FILES/a6/testbooks is a bash script that compiles all the test programs and runs all the supplied test cases, diffing them.

Problem 2. (20 points) SR *[Puzzle Problem]*

The following program, $FILES/a6/SR0.cc, uses a mystery class named SR:

```
#include <cstdio>
#include "SR.h"

int main()
{
    int j = 5;
    int a[ ] = {10, 15};
    {
        SR x(j), y(a[0]), z(a[1]);

        j = a[0];
        a[0] = a[1];
        a[1] = j;

        printf("j = %d, a = {%d, %d}\n", j, a[0], a[1]);
    }

    printf("j = %d, a = {%d, %d}\n", j, a[0], a[1]);
}
```

Here is the output of the program:

```
j = 10, a = {15, 10}
j = 5, a = {10, 15}
```

Here is another example (SR1.cc). The output is "`sum = 161700`".

```
#include <cstdio>
#include "SR.h"
int main()
{
    int sum = 0;
    for (int i = 1; i < 100; i++) {
        SR ii(i);
        while (i--)
            sum += i;
        }
    printf("sum = %d\n", sum);
}
```

Your task is to implement SR. The behavior of your version must match the behavior shown above.

Restriction: Your solution may not contain any asterisks!

Hint: Start thinking about this problem right away!

Problem 3. (10 points) error.cc *[Puzzle Problem]*

The file $FILES/a6/error.cc does not compile. Edit it, adding exactly one line of text, whose presence makes it compilable. **Restrictions: (1) The line can have at most one semicolon. (2) You may not change the accessibility of itsValue—it must remain private.**

Problem 4. (Extra Credit) extra.txt

Submit a plain text file named extra.txt with the following.

(a) (1 point extra credit) Estimate how long it took you to complete this assignment. Other comments about the assignment are welcome, too. I appreciate all feedback, favorable or not.

(b) (1-3 points extra credit) Cite an interesting course-related observation (or observations) that you made while working on the assignment. The observation should have at least a little bit of depth.

Think of me saying "Good!" as one point, "Interesting!" as two points, and "Wow!" as three points. I'm looking for quality, not quantity.

Miscellaneous

You'll have almost three calendar weeks for this one but it's due the Monday after Spring Break. Plan accordingly!

Use diff to BE SURE that the output of your solutions EXACTLY MATCHES the output of the reference versions in $FILES/a6. The set of reference versions and associated data files will freeze exactly two weeks (336 hours) prior to the deadline.

Feel free to use comments to document your source code as you see fit, but note that no comments are required.

You should be able to complete this assignment using the material on slides 1-135.

Don't hesitate to ask me for hints and/or help if you have trouble with a problem. As always, I'll be happy to inspect your solutions for compliance with restrictions before the deadline.

Deliverables

Use `turnin` with the tag 397a_6 to submit your solutions for grading. The deliverables for this assignment are Books.h, Books.cc, SR.h, error.cc, and if you choose to submit it, extra.txt.