Lab 7: Overflowing Bookshelf

Submission

Please submit the pre-lab and the lab as instructed.
Source Code: bookshelf.cpp

Lab Description

Agnes C. Mulligan is a fanatical bibliophile – she is constantly buying new books, and trying to find space for those books. In particular, she has a shelf for her "to be read" books, where she puts her newest books. When she decides to read one of these books, she removes it from the shelf, making space for more books. Sometimes, however, she buys a new book and puts it on the shelf, but because of limited space, this pushes one or more books off the shelf at the other end. She always adds books on the left side of the shelf, making books fall off the right side. Of course, she can remove a book from any location on the shelf when she wants to read one.

Your task will be to write a simulator that will keep track of books added and removed from a shelf. At the end of the simulation, display the books remaining on the shelf, in order from left to right. Books in each simulation will be identified by a unique, positive integer, $0 < I \leq 100$. There are three types of events in the simulation:

- *Add:* A new book is pushed on the left end of the shelf, pushing other books to the right as needed. No book moves to the right unless it is pushed by an adjacent (touching) book on its left. Any book that is not entirely on the shelf falls off the right edge. No single book will ever be wider than the given shelf. No book that is currently on the shelf will be added again.
- *Remove:* If the book is on the shelf, then the book is removed from the shelf, leaving a hole. If the book isn't on the shelf, the operation is ignored.
- *End:* End the simulation for this case and print the contents of the shelf.

Input: The input file will contain data for one or more simulations. The end of the input is signalled by a line containing -1. Each simulation will begin with the integer width of the shelf, s, $5 \leq s \leq 100$, followed by a series of add and remove events. An add event is a single line beginning with an upper case 'A' followed by the book ID, followed by the integer width of the book, w, $0 < w \leq s$. A remove event is a single line beginning with an upper case 'R' followed by the the book ID. Finally, the end event is a line containing only a single upper case 'E'. Each number in an event is preceded by a single blank.

Output: For each simulation case, print a single line containing a label (as shown in the output sample), followed by the list of IDs of books remaining on the shelf, in order from left to right.

Example Input	Example Output
10 R 3 A 6 5 A 42 3 A 3 5 A 16 2 A 15 1 R 16 E 7 A 49 6 A 48 2 R 48 E 5 A 1 1 A 2 1 A 3 1 R 2 A 4 1 A 5 1 R 5 R 4 A 6 1 A 7 4 E -1	PROBLEM 1: 15 3 PROBLEM 2: PROBLEM 3: 7 6

Pre-Lab

Read the complete problem description and then determine what the expected output should be if given the following input:

Pre-LAb Input	Pre-Lab Output
100 A 1 20 A 2 20 A 3 30 A 4 30 R 2 A 5 1 A 6 19 E 7 A 49 1 A 48 1 R 43 A 47 1 A 46 1 A 45 1 A 44 1 A 43 1 A 42 6 R 42 A 49 1 A 48 1 A 47 1 E -1	

Judge's Data

You can run the *automated judge's tests on hopper* to test the correctness of your program (although you must still formally submit the source code via the git system). To use the judging utility, your source code must be named *order.cpp*.

Hints

This was a more difficult programming contest problem than the ones we have seen in our earlier labs. By the end of a five-hour contest, less than 50% of the teams had successfully solved it, and the average solving time for those who finished it was over 3 hours into the contest. That said, it is a rather straightforward problem with proper use of data structures.

The basic simulation sounds most like a queue, with new books placed on one end of the shelf and old books falling off the other end of the shelf. However, the rules also allow Agnes to remove a book from an arbitrary position. So for this reason, we are going to use a list rather than a queue. We will store an iterator for each book that is inserted, marking its position to facilitate a later removal. For each trial of the simulation, we recommend that you declare the following variables:

- **list**$<$ *int* $>$ **bookshelf;**
  IDs for the books currently on the shelf, ordered from left to right. At the beginning of the simulation, the bookshelf is empty.

- **int total(0);**
  The cumulative width of all books that are currently on the bookshelf. Initially, this is zero since there are no books on the shelf.

- **vector**$<$ *int* $>$ **widths(101, 0);**
  An array designating the known widths for books. This will be needed so that when individual books are removed, we can update the total width of the collection. Our syntax automatically constructs a vector with 101 entries, each initialized to width 0. Note that we use 101 so that we can use indices up to 100 inclusive (even though ID 0 is never used in the simulations).

- **vector**$<$ *list* $<$ *int* $>::$ *iterator* $>$ **locate(101, bookshelf.end());**
  A vector of iterators into the bookshelf list. Specifically, locate[k] is an iterator marking the current position of book k on the bookshelf, with bookshelf.end() designating the position of a book that is not currently on the shelf.

Getting Started: To get you going in the right direction, the lab includes *bookshelf.cpp* that includes these declaration in a high-level loop.

Notes of warning:

- When a new book is inserted, this may cause multiple books to fall off the other end. So long as the combined width exceeds the shelf's width, you must keep popping old books off the end.

- Remove might be called for a book that is not currently on the shelf. In this case, you are to ignore the command (yet not crash).

- When outputting the results for a trial, you must get the spacing perfectly, making sure not to include any spurious spaces at the end of the line. In particular, there must NOT be a space after the colon if the bookshelf is empty. This can be done by printing the prompt and the colon, but no other space yet. Then, iterate through the list printing out a space followed by a book ID for each entry of the list. For example, in the first sample output, it might be that you print the line piece-wise as "PROBLEM 1:" + " 15" + " 3" + endl.