Slide 1

Administrivia

- Reminder: Homework 8 due today.
- Only about half of you have turned in a final version of Homework 7. If you haven't, don't forget?
- Homework 9 on the Web. Tentatively due last day of class, but actual deadline will be sometime during finals. In terms of learning, this one is more important than Homework 8, so if you only have time to do one of them, do this one.

Slide 2

Sorted Linked List Example, Continued

- (Review code.)

Slide 3

One More Useful Tool — `valgrind`

- The downside of managing memory with `malloc` and `free` is that getting it right is not easy, and sometimes problems don't show up right away.
- `valgrind` can check for a lot of potential errors, including errors in use of `malloc` and `free`.
- Compile with `-g` and `-O0` and

  `valgrind` executable-name

  (The Makefile for the example shows how to remake the executable with these flags.)

Slide 4

Homework 9 — Binary Search Trees

- Those of you taking CS2 I think have seen these recently? For those of you not in CS2, the assignment references a Wikipedia article; I will also try to post a supplemental video lecture soon. (Dr. Lewis has video lectures about BSTs for his CS2, but he approaches them from an angle I don't really want to use for this course.)
- Homework 9 partially defines an implementation of BSTs in C — declares some functions and includes a test program — and asks you to complete it,
- Review/summary of concepts on next slides.

Slide 5

Binary Search Trees — Definitions

- Trees are a special type of "graph" (collection of nodes connected by edges), in which one node is called the root and has any number of outgoing edges but no incoming edges, and other nodes have one incoming edge and any number of outgoing edges. Each node can store some data.

  Useful for representing hierarchies of various kinds (e.g., the files/directories in a file system).
- "Binary trees" are trees in which each node has at most two children.
- "Binary search trees" are binary trees where the data is something that can be ordered, and for each node, everything in its left subtree is "smaller" while everything in its right subtree is "larger". This makes them good for storing a sorted collection that needs to grow/shrink.

Slide 6

Binary Search Trees — Operations

- With all trees, various kinds of "traversal" (visit all nodes) are possible. For BSTs, "in-order" (left subtree, then root, then right subtree) gives you the data in sorted order (why?). Easy to describe recursively; without recursion pretty tricky.
- "Insert" is not too hard and can be described recursively: Inserting into an empty (sub)tree just means adding the thing to insert as the root node.
- "Find" is also not too bad and easy to describe recursively.
- "Remove" is significantly more difficult: Some cases are easy (removing a leaf), but the worst case, removing a node with two children, is tougher. What works is to replace the node to remove with either the largest element of its right subtree or the smallest element of its left subtree.

Slide 7

Minute Essay

- Have you seen binary search trees in another course? (Really I'm kind of asking you to remind me whether you have taken or are taking CS2.)
- I'm kind of bothered that so many people haven't finished an assignment due two weeks ago (Homework 7). If that's you, can you say what the hold-up is?
- Next time (last class!) I will try to show a few examples of using interesting non-standard libraries and extensions (e.g., for multithreading). Anything else you'd like to hear about?
- Best wishes for a good holiday next week!