Slide 1

Administrivia

- Reminder: Homework 8 due today. If you're not done, send me what you have and try to send a more-complete version later. But . . .
- One more homework (Homework 9). Due last day of finals *and not accepted late*. 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.
- Reminder: If you want (virtual?) attendance points for the days for which lectures were on video, send me a minute-essay e-mail.

Slide 2

Sorted Linked List Example, Continued

- (Finish code — insert, remove, "remove all".)

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; the video lectures for Dr. Lewis's CS2 are also good, and I'll add a link to them soon.
- 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

Computer Representation of Data of Numeric Data, Revisited

- Many (most?) languages strictly define sizes of data types. C defines only minimum ranges. Why?? to allow implementations to do whatever is most efficient, while allowing programmer to make *some* assumptions.
- Example program `sizes.c` gives different answers on a 32-bit system!

Slide 8

Minute Essay

- 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?