Slide 1

Administrivia

- Homework 7 on the Web; due in two weeks. This one also may take you some time, but it will give you more practice with pointers, in the context of a linked data structure.

Slide 2

Minute Essay From Last Lecture

- It sounds like many people found Homework 6 challenging (but it was meant to be).
- The goal was for you to understand pointers better, and some said it helped, others I'm not sure. One person said that so far it had "elucidated [his] lack of knowledge" — I hope that changed.
- Some students apparently had some trouble understanding what the program was supposed to do. "Hm!"?
- Note that the way the program asks you to sort text isn't maybe the one most people would come up with first, nor is it how you'd likely to it in a language that gives you more — but I say mimicking how they work is a lot of trouble in C, so better to try something different.

Slide 3

Sorted Linked List Example, Continued

- (Finish most code — insert, print, "remove all". Completed version online after class.)

Slide 4

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 5

Binary Search Trees — Review?

- In general, "trees" in computer science are a special type of "graph" (collection of nodes connected by edges). In a tree, one node is called the root and has any number of outgoing edges but no incoming edges; other nodes have one incoming edge and any number of outgoing edges. Terminology — parent, child, leaf node. Each node can store some data.
- They're 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 much more difficult — some cases are easy (removing a leaf), but the case of 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? (I think they're usually in CS2 and possibly in Discrete Structures or Functional Languages as well.)