Slide 1

Administrivia

- Reminder: Homework 8 due today. No more required homework (I had thought one more, but no). I will (try to) put together a set of extra-credit problems.
- Reminder: Quiz 6 Wednesday. Likely topic is `struct`s.

Slide 2

Lists and Other "Collection" Types

- We know about arrays as one way to represent a collection of data.
- We could abstract from this a bit and talk about "lists" (what the textbook calls "linear lists") as linear ordered collections of data.
- We could also consider coming up with ways of representing non-linear collections such as trees, graphs (in the sense of a collection of nodes and edges), etc.
- Many/most programming languages support this idea, sometimes through fairly extensive libraries. C, not surprisingly, doesn't, but you can build your own, typically using `struct`s and pointers.

Slide 3

Linear Lists

- One way to implement a linear list is with an array. Simple and efficient if list can be of fixed size and you don't need to add/insert elements in the middle.
- If that doesn't work well, an alternative is a "linked list" consisting of a collection of "nodes", each consisting of a list element plus a pointer to the next element.

Slide 4

Linked Lists in C

- Defining a `struct` for the nodes of a linked list is somewhat tricky in C because one of the fields needed is a pointer to something of the same type. But the following works to define a linked list of `int`s:

```
typedef struct int_list_node {
    int data;
    struct int_list_node * next;
} int_list_node_t;
```

- (Example code for sorted list.)

Minute Essay

- Can you think of situations in which linked lists could be useful?