Slide 1

Administrivia

- Homework 7 due date extended to last day of class, but accepted without penalty through the next Tuesday.
- Homework 8 on the Web; also due last day of class but accepted late.

  This one is actually more important than Homework 7, so if for somem reason you can't do both, do Homework 8.

Slide 2

Homework 7 — Conway's Game of Life

- Many years ago mathematician John Conway came up with a simple zero-player "game".
- Basic game setup is a 2D grid of cells, each of which can contain an organims, or not.
- "Play" consists of updating this grid of cells according to some simple rules. No input to the game after the initial setup (which cells are full/empty, i.e., live/dead).
- Details in Homework 7, and/or the Wikipedia article looks good.

Slide 3

Homework 7 — Conway's Game of Life, Continued

- A program to "play" this game is not so very hard to write, and it's a nice use of 2D arrays;
- Basic calculation is fairly straightforward: Keep two copies of the "board". At each step, for each cell in the board compute its next value based on its current value and current values of neighboring cells, producing an updated board. Then copy this updated board to the first board and repeat.
- Getting initial values for the board and displaying it can be non-trivial: GUI seems best, but beyond this course, and even a text=based interface is tedious, at least for input.
- Homework 7 gives you starter code and asks you to make some I-hope-not-daunting changes and additions, as a way of getting practice with 2D arrays.

Slide 4

Homework 8 — Binary Search Trees

- Dr. Lewis tells me that he covered BSTs in CS2 this week, which I hope means all of you have seen them? (If not, the assignment references a Wikipedia article, or I can talk with you about in office hours.)
- Homework 8 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

- Recall discussion last time of computer representation of integers and reals (the latter as "floating point").
- 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

Computer Representation of Text

- We talked already about how "text strings" are, in C, arrays of "characters". How are characters represented? Various encodings possible.
- One common one is ASCII — strictly speaking, 7 bits, so fits nicely in smallest addressable unit of storage on most current systems (8-bit byte).
- Another one is Unicode — originally 16 bits (Java's `char` type), now more complicated. (Again, Wikipedia article seems okay.)
- Either encoding can be considered as "small integers".
- C's `char` type often ASCII but doesn't have to be. (Older systems use(d) EBCDIC, an encoding rooted in punched cards.) C also has `wchar_t`, which *could* be Unicode.

Slide 9

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?