Slide 1

Administrivia

- Reminder: Homework 4 due *next* week.
- Note that I won't be replying much to video-quiz responses — it's just too overwhelming — though on occasion I will. (My thinking is that for questions with some notion of a right answer, slide after question(s) will have my answer(s), so you should have some idea whether what you said was right.)

  (By the way: Most people are doing great on subject lines for these things. It really does help me, so thank you!)
- For homeworks, I do try try to comment not only on whether what you sent me was right but also on anything really noteworthy (good or bad). But if the only thing I say about a program is "AOK", this is good! it means as I was grading I was probably thinking "nice competent job; nothing to say here".
- Sample solutions for Homework 1 and 2 posted.

Slide 2

A Few More Words About `vim`, Etc.

- Most people seem to have learned something from the tutorial. Good! `vim` is painful to use if you know only the bare minimum but starts to seem reasonable when you know more.
- I have some notes and tips on `vim` under "Useful links" on the course Web site. (Look at it briefly.)
- In case it's not clear, I advocate learning either `vim` or `emacs` but not both, if only so you know a nice lightweight editor that works in a text-only remote session. Which one? several faculty use `vim` and can help you with questions. Both fine editors though! (This used to be a religious-war topic. !?)

Slide 3

A Few More Words About `vim`, Etc.

- If you find beeps annoying, these notes say how to stop them in `vim`. To also stop them on command line, make file `.inputrc` in home directory with lines:

```
$include /etc/inputrc
set prefer-visible-bell
```

Slide 4

Recap of Video Lectures (Group 02 — week of 1/29)

- Using `make` to compile.
- Conditional execution in C.
- `scanf` and checking for errors.
- Recursive functions.
- Library functions in C.
- Questions?

Slide 5

Simple I/O, Revisited

- Doing a really good job with interactive input surprisingly tricky — what constitutes error, how to prompt user to try again.
- So for this class, focus on some simple safety checks: if input should be numeric it is, and values make sense for program (e.g., inputs to GCD program not both 0). I like to always print input values so users can at least confirm that what they thought that typed in is what program read.
- Some online sources discourage use of `scanf`. Other ways possible, and arguments can be made for them, but they have their problems too. Annoying that it doesn't detect overflow, but oh well.
- For this class, best to just bail out on bad input, rather than retrying. (And if you do anything else on homework, it breaks my semi-automated testing.)

Slide 6

Recap of Video Lectures (Group 03 — week of 2/05)

- Loops in C (`while` and `for`).

  Most people's answers to quiz question about printing powers of 2 fine. Note that I recommend *not* using `pow()` to compute integer powers of integers: It converts to/from `double`, with possibly loss of precision, and may be less efficient.
- Arrays in C.

  Why no checking of array indices . . . Partly an efficiency measure, but also no way to do it in general without storing length with array.
- A little about "random" numbers, plus a digression about TEX.
- Questions?

Slide 7

`main()` Revisited

- If no access to command-line arguments needed, declare `main()` as

  `int main(void)`

  Note that in C this is subtly different from

  `int main()`

- Return value from `main()` should be zero if program "worked" (whatever that means in context), something else if it didn't. Appropriate values for the "something else" somewhat implementation-dependent. Nice touch to `#include stdlib.h` and use

  `EXIT_SUCCESS` and

  `EXIT_FAILURE`.

Slide 8

Recap of Video Lectures (This Week)

- An introduction to pointers — perhaps most difficult topic this course addresses, and one that matters most in preparing you for Data Abstraction.
- Strings in C, and how they're different from strings in many other languages. As with so many things in C, interface is a thin veneer over the implementation.
- Command-line arguments in C.
- Questions?

Slide 9

Homework 4

- Second problem gives many students trouble.
- Outline of what you're supposed to do:

  Generate N "samples" using `srand()` and `rand()`.

  Map each to range from 0 to $B - 1$, where B is a number of "bins", and count how many fall into each "bin".

  Starter program has code to do both mappings, so no need to struggle with that.

Slide 10

Minute Essay

- What are you doing about readings for the course? i.e., did you buy a copy of the recommended book, or are you relying on the online tutorial?
- Any questions?