Slide 1

Administrivia

- Reminder: Homework 7 due Friday.
- Readings updated (as mentioned in e-mail).
- Sample solution posted for Homework 6 (of course, don't peek if you're still working on the problems).

Slide 2

Minute Essay From Last Lecture

- Many people had trouble with Homework 6. I do mean for assignments to be somewhat challenging, but I don't mean for them not to be doable with a bit of effort.
- At least one person mentioned how much difference memoization made to the recursive Fibonacci function. One of my original motivations for assigning this problem was to have an incentive to try it myself!

Slide 3

Homework 7

- For the first problem, do try to process input a character at a time. (It's kind of a more interesting logic puzzle that way anyway.)
- For the second problem, you may have to try various inputs to get ones that produce interesting-to-display results.
- Be advised that ACM tutors may or may not know about `gnuplot`, though I think with what's in the homework writeup and the example input file you should be okay — but if not, ask me, perhaps by e-mail.

Slide 4

Just For Fun — ASCII Art Revisited

- I mentioned an all-ASCII-art animation of the first Star Wars movie? apparently still available, via

  ```
  telnet towel.blinkenlights.nl
  ```

  (to interrupt control-] then control-d).
- (What some people choose to do with their time can be — interesting?)

Slide 5

Text Data — Single Characters

- `char` is considered an integer type and can be worked with as such. Notice that while these days ASCII is by far the most common encoding, standard doesn't require that, and there are other possibilities.
- Many library functions for working with single characters (e.g., `isalpha`).
- Character literals represented using single quotes.
- Can read in / print single characters with `scanf` or `printf` using `%c`. Or can use `getchar`, `putchar`. Notice that `getchar` returns an `int`. Why? so it can return special value `EOF` when no more input.

Slide 6

Text Data — Strings

- Most more-recent languages have nice ways of working with "strings" of text data that hide details and provide nice functionality.
- C, in contrast, provides a bare-bones version, in which text strings are represented as arrays of `char`, with an end-of-string character (`'\0'`) that allows an array of fixed size to store strings of different sizes.

  Simple but subject to all the perils of arrays!
- String literals represented using double quotes. Can include "escape" characters (e.g., `'\n'`.)

Slide 7

Text Strings — Output

- Can use `printf` with `%s`.
- Can also use `puts` (which adds a newline).

Slide 8

Text Strings — Input

- Surprisingly (or not, given C's minimalist implementation of arrays), no nice way to do this!
- Can use `scanf`, but no nice/general way to be sure you don't overflow array, and getting something that includes whitespace may be tricky.
- Can get a whole line with `fgets`, but must provide a fixed-size array (so, what size?) and deal with newlines.
- `gets` looks useful but observe what its `man` page says(!).
- Consider processing data character by character, or using command-line arguments.

Slide 9

Working With Text Strings in C

- Once you have some "strings" in your program, what can you do with them?
- You can work on them as arrays of character (that's what they are) or using pointers (as in the example last time with an array of `int`s).
- Perhaps surprisingly, normal(?) assignment and relational operators don't for the most part work, but there are library functions (next slide).

Slide 10

Working With Text Strings in C, Continued

- Many library functions useful for working with strings, among them `strcmp` to use instead of relational operators.
- Significant problem in working with strings — no natural maximum size, so must decide how big to make the array of characters used to hold one — and then be sure you don't try to put in too many characters.
- Some library functions let you say how big the array is; some don't. *Always* be as careful as you can when working with strings; trying to store a string in an array not big enough is a source of "buffer overflows", which can lead to program crashes and more subtle problems, including security risks.

Slide 11

Working With Text Strings in C, Continued

- Many library functions for working with strings use/return pointers. Pointer arithmetic allows for some interesting uses of these functions.
- (Examples as time permits.)

Slide 12

Minute Essay

- None really — sign in (unless questions?).