Slide 1

Administrivia

- Reminder: Homework 3 due today.
- Quiz 2 next Monday.
- Midterm a week from Friday (10/23). (Okay?)
- Next homework to be on the Web soon (today or tomorrow); due next Wednesday.
- If you were in class Friday but didn't send me a minute essay, please do so! this is what I use in calculating the attendance part of your grade.

Slide 2

Recursion — Review

- (Review notes from last time.)
- (Additional example(s)?)

Repetition Via Loops

Slide 3

- Recursion provides one way to repeat something. Often not efficient (every call to a function requires space for local variables, and at some point you can run out of room), nor is it always convenient (writing a function every time you want to repeat something).
- Hence C, like most procedural languages, offers constructs called *loops*. All have four basic elements (sometimes implicit).

Loop Elements

Slide 4

- Initializer — something that sets initial values for variables involved in the repetition (iteration).
- Condition — something that determines whether repetition continues. Can be tested at the start of each iteration (*pre-test* loop) or at the end (*post-test* loop).
- Body — the code to repeat.
- Iterator — something that moves on to the next iteration.

Slide 5

while Loops

- Probably the simplest kind of loop. You decide where to put initializer and iterator. Test happens at start of each iteration.
- Example — print numbers from 1 to 10:

```
int n = 1;                /* initializer */
while (n <= 10) {         /* condition */
    printf("%d\n", n);    /* body */
    n = n + 1;            /* iterator */
}
```

- Various short ways to write `n = n + 1`:

```
n += 1;
n++;
++n;
```

What do you think happens if we leave out this line?

Slide 6

for Loops

- Probably the most common type of loop. Particularly useful for anything involving counting, but can be more general. Syntax has explicit places for initializer, condition, iterator (so it's less likely you'll forget one of them).
- Example — print numbers from 1 to 10:

```
int n;
for (n = 1; n <= 10; ++n) {
    printf("%d\n", n);
}
```

- Initializer happens once (at start); condition is evaluated at the start of each iteration; iterator is executed at the end of each iteration.

`do while` Loops

- Looks very similar to `while` loop, but test happens at end of each iteration.
- Example — print numbers from 1 to 10:

```
int n = 1;                        /* initializer */
do {
    printf("%d\n", n);            /* body */
    n = n + 1;                    /* iterator */
} while (n <= 10);                /* condition */
```

Loop Examples

- (Next time . . .)

Minute Essay

- None really — sign in, unless questions about recursion?