Slide 1

Administrivia

- Reminder: Homework 4 due at 5pm today.
- Reminder: Quiz 4 Tuesday. Likely topic is functions.

Slide 2

Homework 3, Revisited

- Second problem asked you to write a program to solve a quadratic equation

$$ax^2 + bx + c = 0$$

using the rule that if

$$\sqrt{b^2 - 4ac} \geq 0$$

there are two roots given by

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- What happens if $a = 0$? (Oops. See updated solution on Web.)

Slide 3

Functions — Review

- Functions are somewhat like math functions — zero or more inputs, one output (*return value*) or none.
- *Defining* a function – specify
  - Its name (same rules as for variables — letters, numbers, underscores).
  - What parameters it needs (types, "local names" — `void` if none.
  - What type of thing it returns (`void` if nothing).
  - Some code. Can include *local variables*. If function returns something other than `void`, must include at least one `return` followed by the value to return.
- *Declaring* a function — just give name, parameters, return type. Definition can be somewhere else in the program.

Slide 4

Functions — Review, Continued

- *Calling* a function — give its name, values for parameters. This is an expression (in the same sense as, say, `x+1`) and — unless the function returns `void` — has a value, which can be assigned to a variable, used as part of a boolean expression for conditional execution, etc.
- Since a function call is an expression — when we come to one, we evaluate it:

  Pause what's currently happening. Copy values of input variables to function's parameters. Execute code in function until we get to a `return`, or the ending curly brace. Whatever expression follows `return` is the function's (return) value. Continue execution in "caller" using return value.

  Notice that executing code in the function may produce "side effects" (e.g., printing something).

Slide 5

Repetition — Review/Recap

- Several ways to repeat something — recursion. loop constructs discussed last time.
- Which to use? in general, the one that makes the programs easiest for humans to understand — worry about efficiency only when it matters.

Slide 6

Example of Using Loops

- Look again at the "convert English to metric" example program. We could make some improvements (or changes anyway) . . .
- First change it so it lets you do multiple conversions without running the program again.
- Now change the function that gets a number so if you type in something other than a number, it asks again.

Slide 7

Another Loop Example

- We could write the following to print values 0 through 9:

```
int i = 0;
while (i != 10) {
    printf("%d\n", i);
    i += 1;
}
```

- So if we wanted to print values 0.0 through 0.9, we might write

```
float f = 0;
while (f != 1.0) {
    printf("%f\n", f);
    f += 0.1;
}
```

Let's try it . . .

Slide 8

Homework 3, Revisited

- Now that we've been reminded that we can't represent all decimal fractions in floating-point:

  In the first problem (computing income tax), using floating-point numbers of any kind to represent money is a bad idea! commonly done, but — at best sloppy, and sometimes you get answers you don't expect!

- What to do instead? Here, makes sense to compute pennies and then round to whole dollars.

  (Updated solution on Web.)

Slide 9

Minute Essay

- Write a loop to print the even numbers from 0 through 10. (You can use `while`, `for`, or `do while`. If you have time, try doing all three.)

Slide 10

Minute Essay Answer

- Here are some solutions;

```
int n;

n = 0;
while (n <= 10) {
    printf("%d\n", n);
    n += 2;
}

for (n = 0; n <= 10; n += 2) {
    printf("%d\n", n);
}

n = 0;
do {
    printf("%d\n", n);
    n += 2;
} while (n <= 10);
```