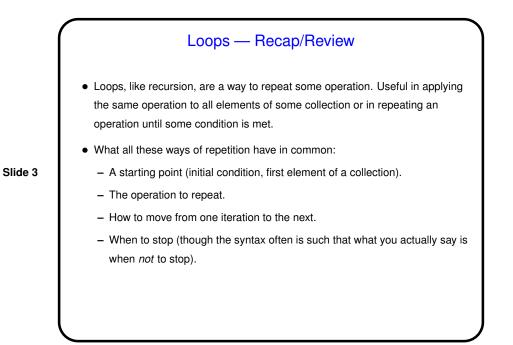
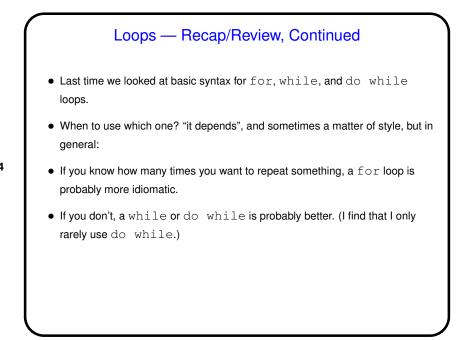


## Minute Essay From Last Lecture

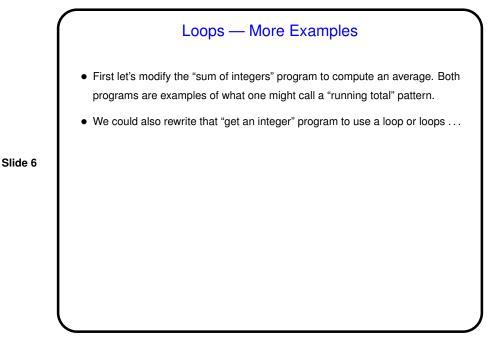
- Everyone had seen loops in some other context, though some said they weren't comfortable with them or didn't remember much. We'll use them a lot!
- One person asked about when to use while and when to use for. In C you can (almost?) always use either one, so it's kind of a matter of style, but ... A little more today.





## Loops Versus Recursion

- As noted in class, recursive functions can be simple to write but potentially inefficient (though in some cases a sufficiently smart compiler can reduce or eliminate the inefficiency look up "tail recursion" to find out more).
- For other problems, a loop is simpler to write loop versions of many of the in-class examples of recursion are as simple or simpler, and that program to get an integer from input without using scanf would have been much simpler with loops. So it may seem that loops are better.
- But there problems for which recursive solutions are much simpler to write and get right, while non-recursive solutions are decidedly not simple anything involving "trees", plus at least two widely-used algorithms for "sorting" (putting things in order).



## A big use of computers is in solving (exactly or approximately) mathematical problems — "numerical computation" or "numerical analysis". Matlab is one tool for this, and/or you can write your own programs in a general-purpose programming language. Often (maybe always?) these involve various forms of repetition. An example is "numerical integration", in which you approximate a definite integral (area under a curve) by computing areas of rectangles and adding them up. As an example ... (next time).

