Why do we love to program?

- The “sheer joy of making things”
- The joy in “making things that are useful to other people”
- The “fascination of fashioning complex puzzle-like objects of interlocking parts and watching them work in subtle cycles, playing out the consequences of principles built in from the beginning”
- The “joy of always learning”; here, “the problem is ever new”
- The “delight of working in such a tractable medium”
- The flexibility through which we create, polish, and rework
What does it mean to scale up?

- Fundamentally, to scale up is to increase something in size or number or by a specific ratio.
- We often ask whether a given system or solution can successfully scale up to meet larger demand in terms of numbers of inputs, users, etc.
  - How can we be sure?
  - How can we test a system to be sure?
- Pattern: we identify a bottleneck, rework that part of the system and fix the bottleneck, but then introduce a new bottleneck elsewhere within the system; repeat!

Why is scaling up so difficult?

- a simple program (usable by its author) → 3 times the cost → 3 times the cost → a programming product (can be run, tested, extended by anyone) → 9 times the cost → a complete programming system and product (a fully tested and fully operational large-scale system) → 3 times the cost → a programming system (a collection of interacting and fully coordinated programs)
Why is scaling up so difficult?

- “Human beings are not accustomed to being perfect, and few areas of human activity demand it.”
- Perfection is required when programming; or is it?
- “All programmers are optimists” (but we’ll never find the last bug!)
- “[L]arge programming projects suffer management problems different in kind from small ones, due to division of labor.”
- Often, “other people set one’s objectives, provide one’s resources, and furnish one’s information.”
- We rely on the code, libraries, and interfaces of others.

Why is scaling up so difficult?

- There is an “unvoiced assumption...that all will go well.”
- “Our estimating techniques fallaciously confuse effort with progress”; further, “schedule progress is poorly monitored.”
- The time and effort required to communicate is often overlooked, as is the time required for training and research; (what else?)
- “[T]esting is usually the most mis-scheduled part of programming.”
- Brooks uses this breakdown:
  - “1/3 planning; 1/6 coding; 1/4 component test and early system test; 1/4 system test, all components in hand.”
Coding in any language

- What are best practices for writing “good” code?
  - Document your code using comments
  - Use proper and consistent indentation
  - Limit line lengths and source file sizes
  - Establish consistent naming conventions
  - Minimize scope to the extent possible
  - Keep it simple
  - And be sure to always feed your developers all sorts of free food!

For next time....

- Write perfect C code to solve the three problems below.
  - You are not allowed to use any library functions from string.h
  - Bring a printed copy of your solutions to our next class

- Write a function to determine whether a given string called needle is a substring of another string called haystack

- Write a function to count how many times a given string called needle is a substring of another string called haystack

- Write a function to determine the longest palindrome in a given string called wow