QUIZ SECTION EXERCISES

Quiz Section Exercises FAQ

- What is the point of these assignments?
  - Removing some of the nice/safe assumptions that you get to make with your projects
  - Introducing contradictory/impossible requirements
  - Forcing design decisions around complications
  - Showing how early design choices impact later development

FAQ, cont'd

- Why did I lose points for:
  - An algorithm choice
    - Does it run in the required Big O time? In the worst case?
    - If not, did you justify your choice?
  - A design choice
    - Is there any organization or structure?
    - Will it make future changes harder? Is it brittle?
  - A documentation choice
    - Did you make any statements that were factually inaccurate? Eg, put the wrong runtime for an algorithm
    - Did you make a nonstandard choice without justification or explanation?
    - Did you contradict an assignment requirement and not explain it? Eg, Big O of space not runtime

FAQ, cont'd

- What do I need to document?
  - Nonstandard practices
  - Nonintuitive behavior
  - Assumptions
  - Nonobvious boundary cases or breaking behavior
  - Decisions that go against assignment requirements/guidelines
- How much do I need to write?
  - As much as it takes to document your choices.
  - Does not have to be long! Just has to be complete.

URL Validation

- Design is due Nov 14
  - The *only* code change for this assignment is adding an argument to the command line or interface that specifies a sort of: valid URLs, invalid URLs, or all URLs.
- Code is due Nov 21

URL Validation

- What is the right approach to this?
  - Think about time constraints
  - Consider what your common cases are
  - Remember you're free to make assumptions as long as you define what they are. *You should also be able to justify them*.
- A simple approach is fine.
  - http://en.wikipedia.org/wiki/URL_normalization
  - The normalizations that preserve semantics are easy. These are a good place to start.
  - Probably at least 2-4 of the normalizations that may not preserve semantics would make this much more useful.
  - Look what regex exist for validation and build from there.

URL Validation, design

- Code needs to include:
  - Validator
  - Normalizer
  - Comparators
- URL object would let you override comparators. This is good for sort and might be convenient.
- You can leverage parts of this to simplify others. Comparison is easier (and works better) on normalized URLs. It might also simplify validation.