Software Review

* Shell Commands
  + Ssh
    - secure shell, a way of logging into your server from a remote computer (laptop, desktop)
  + Scp
    - secure copy file
  + Valgrind ./a.out >&error.txt
    - Able to read first error easily
    - Stdout – giving output to a file
    - Stderr – standard error, output to the command window
  + Which
    - Prints out full path of output
  + Wc
    - Word count
  + Wget
    - Downloads file into working directory
  + Curl
    - Transfer files, and outputs code for webpage
  + Grep
    - “global regular expression print”
    - processes text line by line and prints any lines that match a pattern
    - used for searching text
* SCM
  + Source Code Management System
  + Example; GIT
  + Features
    - Version control, conflict management
  + Traditional SCM and Git
    - Centralized repo versus distributed repo
  + Git objects
    - Tree, commit, tag
  + Git commands and target areas
    - Remote and local repo, working tree
    - Git push – pushes new content to repo
    - git pull – retrieves new content (refresh)
    - git clone - downloads whole repository
    - git fetch – downloads object from other repository
    - git commit – comments what was changed
    - git checkout – update HEAD and set branch as current
    - git add – adds a change to working directory
    - git init – creates an empty git repo to make new one
* cppcheck
  + static analysis tool
* doxygen
  + documentation tool
* ArgoUML
  + UML tools
* VirtualBox
  + Virtualization
* Valgrind
  + Dynamic Analysis Tool
* CQRS
  + Design pattern
* Software Engineering Life Cycle, Deliverables, and Tools
  + Requirement capture and analysis – open-source programs, data to collect, interface requirements
  + System analysis and design – dynamic, static, testing framework
  + Implementation – sublime, VisualStudio, eclipse
  + Testing – valgrind, cppcheck, googletest
  + Deployment – github, ssh
* Target problems
  + Multiple engineers need to modify the same set of source code (git)
  + Multiple versions of the same software needs to be maintained (git)
  + How to run unit testing cases automatically? (google test framework)
  + How to ensure that programmers will actually create code

documentation? (Make it easier for them.) (Doxygen)

* + How to ensure that programmers will indeed update documentation?

(Make the doc live side by side with code.) (Doxygen)

* + How to avoid typing the gcc commands with many parameters over

and over (and watching for the timestamps)? (Build a Makefile.)

* + Memory leaks (Valgrind)
* Tool strength
  + Git: performance in branch management
  + Git: decentralized repo
  + VirtualBox: available on all platforms
  + Google Test Framework: test automation, Mock framework for mock classes.
  + Doxygen: doc generation for multiple languages
  + Github: Rich web interface to git repo
  + Valgrind: no re-compilation necessary
  + Make: dependency-based selective recompilation
  + Shell: command line access to OS functionalities
  + Shell: everyday task automation
* Tool weakness
  + Git: troubleshooting for more advanced opetations
  + VirtualBox: resource requirement
  + Google Test Framework: large code base, generated Makefile
  + Doxygen: most command-line based tools
  + Github: commit comments only available on the web, no approve features
  + Make: hard tab
  + Valgrind: memory usage
  + Shell: many things to memorize