Lecture 1. Introduction to Software Testing

- More theoretical
  - More about "looking at code, rather than editing it"
- Have a look at worksheets before coming in
- Mid-sem is planned for lecture slot
- Same as final exam
- Groups for assignment are self-picked
  - 3 or so people
- Idea for assignment:
  - Find some code (open source, projects)
  - test ~~it~~ it
  - Look at how it was tested vs how it should have been tested
- Why Test?
  - Ideally you should test before even writing code
  - Or, designing test
  - "If you don't start planning for each test when the functional req.s are made, you'll never know why you're conducting the test"
  - consider what fact each test is trying to verify
- More on V&V (I)
  - Validation: "does the software meet the user's needs?" / "are we building the right software?"
  - Verification: "does the software meet the requirements specifications?" / "are building the software right?"
- More on V&V (II)
  - Validation: usability testing, feedback

- Verification: unit testing / inspections static analysis
- Static and Dynamic Testing
  - Static: testing without execution
    - Includes software inspections and some forms of analyses
    - Very effective at finding certain kinds of ~~bugs~~ problems - especially "potential" faults (problems that could lead to faults when program is modified
  - Dynamic: Testing by execution with real inputs
- Software Faults, Errors and Failures
  - #1 source of lost marks
  - Software fault: a static defect in the software
    - "Thing in code you can point at, make a comment about and say 'this part in incorrect and should be fixed'"
  - Software Error: An incorrect internal state that is the manifestation of some fault
    - "An error may result in a failure but it might not."
  - Software Failure: External, incorrect behaviour with respect to the req.s or other description of the expected behaviour
- Testing and Debugging
  - Testing: Finding inputs which cause software to fail

- Debugging: the process of finding a fault given a failure

• Fault & Failure Model

- The 3 conditions necessary for a failure to be observed:
  1) Reachability: the location or locations in the program that contain the fault must be reachable
  2) Infection: the state of the program must be incorrect
  3) Propagation: The infected state must propagate to cause some output of the program to be incorrect