4156 10/27/16

we will not discuss exam until after grading

1st continue black box material, with boundary conditions — discuss "test oracle"

now grey box - still don't read code (maybe)

similar to black box but looks below surface
instead of input/output visible to user,
looks at "internal" I/O visible
w/in system & between (sub) systems

logs & audit trails
saved persistently
recoverable?
correct format & contents
file permissions

data could be wrong for long periods before user visible

data destined for other systems
e.g. database or other
servers on network
check format for all
outgoing data
check data details
including responses

4/56

10/27/16

system-added info
checksums
timestamps

scraps left lying around
security risk & resource leak
delete what is supposed to be deleted
undo vs. expunge
uninstall should leave system clean

think about everything that could go wrong

special case for web browser,
which is "another system"
http headers
cookies
cache expiration

penetration testing
security team can access everything
visible from browser e.g.
javascript, html
& are very knowledgeable about
vulnerabilities

4156 10/27/16

white box testing - leverage inside knowledge of code

"coverage"

exercise all statements
exercise all branches
exercise all def-use paths
exercise all xxx ...

intuition - if you've never executed a given xxx in the program, how can you have any confidence that it works?

basic approach

find a coverage tool for your programming language (or platform)

run your full set of black box & grey box tests

check which xxx still have not been executed & invent new inputs to force them

4156 10/27/16

so what are the xxx? and how do we force?

statements

find out which statements have never been executed by your test cases & why not?

are they unreachable? this should be caught by static analyzer or compiler

if not, what inputs do you need to provide - possibly at full application level, possibly input to enclosing method, possibly return value from method called w/in that method or library/system call

branches- both legs

if neither leg has been executed, then same as statement

if only one leg, what do you need to provide to force other?

4156 10/27/16

in both cases, in order to force a path, might need to trace backwards what a series of conditionals (possibly including loop conditions) need to be
→ "path conditions"
and then solve the constraints

besides statements & branches, which are the most common coverage metrics, other xxx might include def-use paths

definition = write to variable
use = read from variable
similarly need to trace paths
some definitions may be "dead" - never used

also valuable to consider all error handling, either using conditions or exceptions, but may be difficult to force all situations

does invalid data generate appropriate error (not just any error)
no uncaught exceptions

4156 10/27/16

other white box concerns besides coverage

does code clean up after itself?
e.g. release resources -
memory, file handles,
synchronization variables,
db & net connections

also consider what code will do when it cannot get needed resources

thread-safe code when multi-threading is used

use of security roles

4156 10/27/16

many test cases can be automated via a testing tool

test tool runs a set of test cases automatically but doesn't write the test cases for you (although there are some automatic test generation tools)

useful for continuous integration — test after every commit or at periodic intervals such as nightly

* also useful for "regression testing" new feature or bug fix may introduce (other) bugs in code what did you break?

test cases should be *independent* run in any order get same results no dependencies - setup/teardown no "flaky tests"

→ for expensive test cases, where same state or resource is needed, combine into same test class or group