SE 6367 Final Exam
Topics and Review

Professor Mark Gabel
Department of Computer Science
University of Texas at Dallas
Exam Style

• Much like our quizzes
• True/False w/justification
• Multiple choice w/justification
• Computation (algorithms and techniques)
  – “Math problems” with one right answer, as in the quizzes
• Short response
  – Like quizzes, they will range from “one simple answer” to “requires some thought”
  – Testing *understanding* is the key; memorizing definitions won’t help as much here
• Creative problem solving
  – Longer question(s) that require you to apply the techniques we learned in class creatively
Exam Size

• Time
  – 1 hour and 15 minutes
  – I will give you an extra 5 minutes (5:20 PM end)
  – Show up on time and start immediately; 5:20 is the *strict ending time*

• Length
  – About 3 to 5 quizzes
  – More time constrained than usual
  – Work quickly and trust your instincts
Topics

Disclaimer: everything presented in the class is *fair game* for the exam unless I explicitly stated that a topic would not be tested. The list of topics here is a good-faith attempt at a comprehensive list, but I cannot guarantee it is perfectly complete.
Topics: Testing Fundamentals

• What is possible and what is not
• Correctness, Reliability
• Functional vs nonfunctional testing
• Interpreting requirements for testing
• Input domain
• The components of a test
  – Test Input
  – Test Oracle
• Qualities of a “good” test (completeness, variety, etc.)
• Testing and how it increases “confidence” in quality
Topics: Generating Tests from Requirements

• **Terms**
  – Functional testing
  – Black box testing (vs. white box)

• **Requirements**
  – Use cases
  – The *test selection problem*
  – Challenges of working with input domains
    • Size
    • Complexity

• **Generating Tests**
  – Equivalence Partitioning (what is it, how do we perform it, and why is it important)
  – Multidimensional, Unidimensional Partitioning
  – Boundary Value Analysis
Topics: Coverage-based Testing

• Confidence and its relationship to Coverage
• Definition of a coverage metric *in general*
  – Coverage domain
  – Feasible vs Infeasible elements
• Advantages and benefits of coverage-based testing
• Risks and limitations of coverage-based testing
• The general coverage testing process
  – Define domain
  – Test and measure
  – Enhance tests
Topics: Preliminaries for Code Cov. Testing

• Control flow graphs
  – Basic blocks
  – Nodes and edges
  – Paths
  – Feasible vs Infeasible paths

• The structure of programs
  – Dominance relationships
  – Superblocks
  – *Why all of this is useful and important*
Topics: Test Adequacy and Enhancement

• Measuring test adequacy with Test Adequacy Criteria
  – Black box vs white box criteria
  – Defining and enumerating a criterion
  – The process (define criteria, test and measure, enhance)

• Infeasibility
  – A central problem; know it well
  – Understand why it is a hard problem
  – How one convinces oneself (or proves) that a coverage element is infeasible
Topics: Test Adequacy and Enhancement

• Control-flow based coverage metrics
  – Statement/block coverage
  – Condition and decision coverage
  – Multiple condition coverage
  – Path coverage

• Data-flow based coverage metrics
  – Definitions
    • Definitions, def/use relationships (c-use, p-use)
    • Dataflow graph
    • Def-clear path
  – All-uses metrics (All-C-Uses, All-P-Uses)

• Advantages and disadvantages of various coverage metrics
  – Example: as a coverage metric becomes more “fine” or “detailed”, test suites that satisfy it have more bug finding power. However, feasibility becomes more of a concern.

• Relationships between adequacy criteria
  – Subsumption
Topics: Regression Testing

• Definition of *Regression*
  – Feature regressions (breaking a working feature)
  – Bug-fix regression (un-fixing a bug)
• The regression testing process and its *importance*
• Execution slicing
• Regression test selection
  – how do we decide which tests to run?
• Test set minimization, including our simple greedy algorithm. Why do we need to minimize regression test sets?
Topics: Test Minimization and Coverage Tools

• Test minimization and prioritization
  – Presented in the context of regression testing, but generally useful
  – Know the what, how and why
  – Both need a “metric” to optimize for; coverage is most frequently used.

• Coverage tools
  – How tracing and sampling works
  – The tradeoffs between tracing and sampling
Topics: Mutation Testing

• Process
  – As described in the slides and on our quiz
• Know the purpose of mutation testing
• Definitions
  – Mutant
  – First-order mutant
  – Higher-order mutant
  – Distinguished/Killed and Live mutants
  – Family of mutant
  – Mutation operator (examples too)
Topics: Mutation Testing

• The equivalent mutant problem
  – What is it and why is it hard to solve
  – How does it relate to “feasibility” in coverage-based testing

• Mutation score

• Know about the power of mutation testing as it compares to control-flow adequacy assessment (subsumption, etc.)
  – But specific proofs (in slides) that mutation testing subsumes different types of dataflow coverage are not tested

• Techniques for lowering the cost of mutation testing
  – Weak mutation
Topics: Randomized Testing

• What it is
• What it is good for
• Advantages and disadvantages
• Revisit the importance of test oracles
  – Incomplete oracles
• Why fuzzing a known-good input may be more effective than generating purely-random input
• Once again, be able to think critically about test oracles
  – Very important practical skill
Topics: Debugging – Fault Isolation

• Execution slicing and dicing
  – How to compute
  – What data are needed

• Strengths and limitations
  – Theoretically (e.g. dices don’t necessarily always contain the fault)
  – Practically (e.g. dices can be huge or nonexistent)
Topics: Delta Debugging

• The purpose: minimizing failing test inputs
• Why we simplify inputs
• Know how to perform a binary search over program inputs to find a simpler test case
• Potential applications of delta debugging
• Not tested:
  – Exact details of the more general algorithm
  – 1-minimality
Topics: Statistical Debugging

• Difference between traditional debugging and statistical debugging
  – Main point: the use of both passing and failing runs to automatically isolate faults.

• Tarantula
  – How it works (what data it needs and how it uses it)
  – Its relationship to execution dicing

• Not tested
  – Exact formulas and algorithms from either of the techniques
Topics: Program Analysis and Verification

- Know that they exist
  - Just because something isn’t possible with testing doesn’t mean it’s impossible *in general*
- Study brief overview lecture
  - Understand (but don’t memorize) the definitions