EECS 481 — Software Engineering
Fall 2019 — Exam #1

- **Write your UM uniqname and UMID and your name on the exam.**
- There are nine (9) pages in this exam (including this one) and seven (7) questions, each with multiple parts. Some questions span multiple pages. If you get stuck on a question, move on and come back to it later.
- You have 1 hour and 20 minutes to work on the exam.
- The exam is closed book, but you may refer to your two page-sides of notes.
- Even vaguely looking at a cellphone or similar device (e.g., tablet computer) during this exam **is cheating**.
- Please write your answers in the space provided on the exam. Clearly mark your solutions. You may use exam margins for scratch work. Do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. We may deduct points if your solution is far more complicated than necessary.
  - *Good Writing Example:* Testing is an expensive activity associated with software maintenance.
  - *Bad Writing Example:* Im in ur class, @cing ur t3stz!1!
- If you leave a non-extra-credit portion of the exam blank, **you will receive one-third of the points for that small portion (rounded down) for not wasting time.**

UM uniqname: ______________________________

UM ID: ______________________________

Name (print): ______________________________

1 Software Process Narrative (13 points)

(1 pt. each) Read the following narrative. Fill in each ____ blank with the single *most specific or appropriate* corresponding concept from the answer bank. (Each ____ blank does have a corresponding answer.) Each option from the answer bank will be used *at most once.*

A. Alpha Testing	B. Beta Testing	C. Call-Graph Profile	D. Comparator
E. Conditional Breakpoint	F. Dataflow Analysis	G. Development Process	H. Dynamic Analysis
I. Flat Profile	J. Formal Code Inspection	K. Integration Testing	L. Mocking
M. Oracle	N. Passaround Code Review	O. Perverse Incentive	P. Priority
Q. Quality Property	R. Severity	S. Software Metric	T. Spiral Development
U. Threat to Validity	V. Triage	W. Watchpoint	X. Waterfall Model

A collaborative educational software company is working on a new on-line combined coursework and cuisine forum: *Pizza.*

____ To assess the total market for a potential feature of this cuisine-and-coursework software, management only surveys chefs and forgets to survey students.

____ The customers require the software to be efficient and also to preserve user privacy.

____ Software development is organized around the repeated generation of increasingly-complex prototypes, based on assessments of risk.

____ Developers want to know how much time is spent in a particular function alone, as well as when all of its children are considered.

____ During initial development, other groups within the company are asked to evaluate the software and look for bugs.

____ Before the back-end database is fully implemented, code that depends on it is tested with respect to prototype functions that always return the same answers to every query.

____ When evaluating the HTML output of an interaction, the "current time" field of the produced webpage is not considered.

____ To help find race conditions, developers run the program and track the set of locks held during every memory access.

____ To debug an issue inside a tight loop, developers arrange for execution to be interrupted only on the 10,000th iteration of that loop.

____ A new defect report comes in. QA finds it to be a valid report and not a duplicate. Developers are instructed to address it.

____ The new defect is a memory corruption issue. Management decides it must be fixed immediately.

____ The patch for the bug is shown to two other developers who assess its quality.

____ To favor high-quality code, management institutes a policy in which developer bonuses are awarded for fixing reported bugs. Developers intentionally create new bugs, report them, and fix them.

2 Testing and Coverage (17 points)

(4 pts.) Fill in the blanks in the code below with boolean expressions so that (1) all three parameters (`a b c`) appear at least once in the method body, and (2) any test suite that obtains 100% path coverage of feasible paths requires at least four test inputs.

```
void frances_allen(int a, int b, int c) {

  if (___________________________)       STMT_1;
  else                                   STMT_2;
  if (___________________________)       STMT_3;
  else                                   STMT_4;
}
```

(4 pts.) Write a small method that accepts one parameter x and write a single test input for it such that your single test input maximizes statement coverage but not path coverage.

(5 pts.) Write a small method that accepts one parameter x such that the test $\{ x = 3 \}$ has 50% branch coverage but the test $\{ x = 7 \}$ has 100% branch coverage.

(4 pts.) Consider a modified version of coverage-based fault localization that uses branch coverage counts instead of statement coverage counts as its mathematical basis. Support or refute the claim that it would be a better localizer.

3 Short Answer (18 points)

(a) *(4 pts.)* Choose an example of software development (from your own experience or from class) where "something went wrong" and use that example to illustrate the relationship between *uncertainty*, *risk* and *measurement*.

(b) *(3 pts.)* Support or refute the claim that the number of *duplicate* defect reports for the same issue should influence its assessed *severity*. Describe a situation in which a defect report might be resolved without an associated code patch.

(c) *(3 pts.)* Support or refute the claim that newly-written xUnit-style (PyTest, unittest, etc.) *unit tests* should be required to meet a minimum software *readability metric* value to be checked in to a project.

(d) *(4 pts.)* Identify a developer expectation of modern passaround *code review* that is commonly met. Identify a developer expectation of code review that is rarely met. Describe a buggy patch that code review is unlikely to correctly reject.

(e) *(4 pts.)* Explain how automated whitebox *test input generation* uses *path predicates* and *constraint solving*. Describe two situations where test input generation is unlikely to succeed at producing high-coverage test inputs.

4 Mutation Testing (14 points)

Consider the following implementation of duplicate array element detection and some associated test inputs, oracles and suites:

```
 1 def duplicates(arr):
 2   n = len(arr)                  # len([5,6]) == 2
 3   answer = False
 4   for i in range(n):            # range(3)   == [0,1,2]
 5     for j in range(i+1,n):      # range(2,4) == [2,3]
 6       if (arr[i] == arr[j]):
 7         answer = True
 8   return answer
 9
10 testInput1 = []                 # oracle1 = False
11 testInput2 = [3,3]              # oracle2 = True
12 testSuiteA = [ testInput1, testInput2 ]
13 testInput3 = [4,5,6]            # oracle3 = False
14 testInput4 = [7,8,9,8]          # oracle4 = True
15 testSuiteB = [ testInput3, testInput4 ]
16 testSuiteC = [ testInput1, testInput2, testInput3, testInput4 ]
```

(6 pts.) Suppose the only mutation operator available to you is *Rename*, which renames one use of a variable to another. *Rename* requires 3 parameters, A, B, and C, and renames the first use of variable A on line B to C. For example, you might rename the reference to `answer` on line 3 to `i`. You may only mutate the method body (lines 2–8), not the tests.

Give an example of a single mutation such that *one of* `testSuiteA` or `testSuiteB` kills the mutant *but not both* (with no Python runtime errors, such as references to undefined variables or index out of bounds accesses) — and say which one kills the mutant.

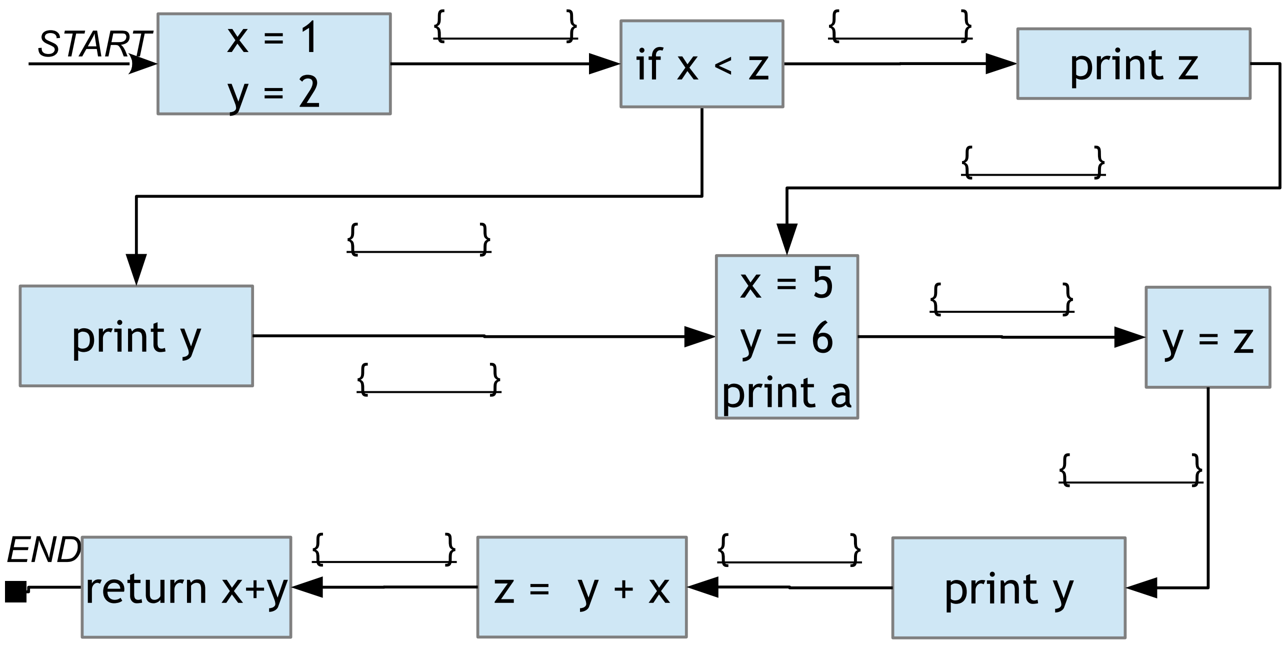
(6 pts.) You are still limited to single variable renaming mutations. Give an example of a single mutation such that both `testSuiteA` and `testSuiteB` kill that mutant (i.e., that mutant fails at least one test in `testSuiteA` and also fails at least one test in `testSuiteB`) without any Python runtime errors.

(2 pts.) Consider a single mutant that changes the `==` on line 6 to `<`. For how many tests in `testSuiteC` does this mutant obtain the wrong answer? What is the mutation adequacy score of `testSuiteC`, expressed as a fraction, with respect to that single mutant?

5 Dataflow Analysis (20 points)

Consider a *live variable* dataflow analysis for *three* variables, `x`, `y` and `z`. We associated with each variable a separate analysis fact: either the variable is possibly read on a later path before it is overwritten (live) or it is not (dead). We track the *set* of live variables at each point: for example, if `y` and `z` are alive but `x` is not, we write { `y, z` }. The special function `print` reads, but does not write, its argument. (You must determine if this is a forward or backward analysis.)

(16 pts.) Complete this live variable dataflow analysis for `x`, `y` and `z` by filling in each blank *set* of live variables.



(4 pts.) Consider the following two *transfer functions* for the "definitely `null`" analysis (sometimes called "constant propagation"). One transfer function handles variable increments, the other handles two joining paths. Name two significant problems that would arise with the dataflow analysis algorithm if we adopted them.

$$\begin{tikzcd} x = i \arrow[d] \\ \boxed{\texttt{x := x + 1}} \arrow[d] \\ x = i + 1 \end{tikzcd}
\qquad
\begin{tikzcd} x = c \arrow[dr] & & x = d \arrow[dl] \\ & x = \max(c, d) & \end{tikzcd}$$

6 Quality Assurance Analyses (18 points)

(3 pts. each) Read the analysis task descriptions. For each task, choose three components of a good analysis solution, one from each column of the table. (For example, "a 1 p" is valid but not "a b 2" or "c 3".) If multiple combinations might fit, choose the *best* answer or the one corresponding to a tool or technique from class or the readings. You may use an option more than once across multiple questions.

a. Automated Dynamic Analysis	1. Enumerate	m. Abstracted Values
b. Automated Static Analysis	2. Replace	n. Common Subroutines
c. Manual Dynamic Analysis	3. Solve	o. Scheduler Interleavings
d. Manual Static Analysis	4. Track	p. Sets of Locks Held

- ___ ___ ___. You wish to determine if a program variable `ptr` could ever possibly take on the value `null` on any run of the program that reaches line 56.
- ___ ___ ___. You wish to determine if there are any shared variables for which a program's threads do use use a consistent concurrency control policy.
- ___ ___ ___. You wish to assess the availability of a service even if its dependencies or assumptions might fail.
- ___ ___ ___. You wish to inspect a small program, for which you have no available inputs, to obtain a human's assurance that it handles critical sections correctly, regardless of the order of its thread operations.

(3 pts.) Support or refute the claim that an effective approach to finding performance defects (e.g., running slowly, using too much energy, etc.) would be to identify frequently-executed regions of code via sampling-based profiling and apply formal code inspection there.

(3 pts.) Support or refute the claim that false positives (i.e., false alarms) are more important than false negatives (i.e., missed bugs) when managers consider deploying bug-finding analyses in software engineering.

7 Extra Credit **(1 pt each; we are tough on reading questions)**

(Feedback) What is one thing you would change about this class for next year? What is one thing you like about this class?

(Free/Participation) Make up one "believable" (but possibly outlandish) claim about the class, professor or course staff. If we get enough interesting responses, we'll post them.

(My Choice Psych) Briefly describe the setup and outcome of Sherif's *Robber's Cave* psychology experiment studying *Realistic Conflict Theory.* (Hint: we played a related musical theatre song in class before discussing it.)

(My Choice Reading) In Terrel et al.'s *Gender differences and bias in open source*, what high-level conclusion was drawn?

(Your Choice Reading 1) Identify any different optional reading. Write a sentence about it that convinces us that you read it critically. (Our subjective judgment applies here!).

(Your Choice Reading 2) Identify any different optional reading. Write a sentence about it that convinces us that you read it critically. (Our subjective judgment applies here!).