# EECS 481 — Software Engineering
# Winter 2020 — 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**.
- Use a **dark pen or pencil** to write your answers in the space provided on the exam. If our scan does not pick up your pen or pencil you will not receive credit. 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 or drawn an X through it, **you will receive one-third of the points (e.g.,** $4/3 = 1.33$**), for that portion 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 | P. Perverse Incentive | Q. Priority |
| R. Quality Property | S. Severity | T. Software Metric | U. Spiral Development |
| V. Threat to Validity | W. Triage | Y. Watchpoint | Z. Waterfall Model |

A software company is working on a new poetry puzzle game based on the works of golden age Tang Dynasty poet Li Bai and 20th-century activist poet Maya Angelou.

____ The company decides to organize its software engineering efforts into distinct phases.

____ To save resources, calls to a prose-fetching library always return the exact same text from *I Know Why The Caged Bird Sings*, rather than the actual prose requested.

____ Developers perform an analysis that they expect will find defects and improve code.

____ After separately constructing an Eastern Poetry module and a Western Poetry module, management instructs engineers to determine if they work together.

____ Before the game is released, some users outside the company are asked to comment on a puzzle related to *A Quiet Night Thought*.

____ Because of context-specific variations in translation, developers arrange for test answers such as "Thoughts in the Silent Night" or "Contemplating Moonlight" to be deemed acceptably close to the reference answer of "A Quiet Night Thought".

____ Developers are concerned about the software's performance on mobile devices.

____ To resolve performance problems, engineers desire a nuanced report that explains the time taken by each procedure and its children.

____ A report related to "moonlight" is received, but it is judged to be too vague to pursue.

____ A defect related to the handling of the *Still I Rise* puzzle is perceived to be of strong relevance to users.

____ To track down a bug related to corruption of the global `stanza` variable, developers arrange to be alerted whenever its value changes.

____ Developers determine the `stanza` bug to be caused by a race condition, so they run the program and track the set of locks held at each access to it.

____ Management awards a monetary bonus to developers for each document written in rhyming couplets, hoping to improve creativity. Developers focus on writing rhymes.

## 2 Testing and Coverage (20 points)

*(4 pts. each)* Consider the following program with blanks. We are concerned with statement coverage, but only for the statements labeled `S_1` through `S_5`.

```
1  void shamir(int a, int b, int c) {
2    S_1;
3    if (a == ______)              { S_2; }
4
5    if (______ == 3)              { S_3; }
6
7    if (a ______ b)               { S_4; }
8
9    if (a ______ c == b)          { S_5; }
10 }
```

Fill in each blank in the program with a **single** letter, integer or operator so that it matches the following coverage results. Here are four test inputs:

| T_1 = shamir(1,2,3) | T_2 = shamir(5,3,2) | T_3 = shamir(5,10,5) | T_4 = shamir(3,3,4) |
|---|---|---|---|

And here are some corresponding coverage measurements for statements `S_1` through `S_5`:

| | | | |
|---|---|---|---|
| $\{$T_1$\} \mapsto 2/5$ | $\{$T_2$\} \mapsto 3/5$ | $\{$T_3$\} \mapsto 4/5$ | $\{$T_4$\} \mapsto 2/5$ |
| $\{$T_1, T_2$\} \mapsto 4/5$ | $\{$T_2, T_3$\} \mapsto 5/5$ | $\{$T_3, T_4$\} \mapsto 5/5$ | $\{$T_2, T_4$\} \mapsto 3/5$ |

*(4 pts.)* Suppose we define full *arithmetic coverage* to require that every arithmetic expression evaluate to both a positive number and a negative number. This is analogous to how branch coverage requires each branch to evaluate to both true and false. Give a smallest test suite for `allen()` below that maximizes *arithmetic coverage*. (Boolean expressions are not arithmetic expressions. Zero is neither positive nor negative.)

```
1  void allen(int a, int b) {
2    int p, q, r;
3    p = a + b;
4    q = a * b;
5    if (a < b)     { r = p * q; }
6  }
```

Answer:

# 3 Short Answer (17 points)

(a) *(4 pts.)* In Dr. Leach's guest lecture, one claim considered was that certain development processes can make creative activities, such as research, inefficient. Support or refute that claim using two examples.

(b) *(6 pts.)* Your company wants to build high-quality software quickly and inexpensively. The company currently does not use static analysis and is considering re-allocating 10% of its testing effort/budget to static analysis. Identify two *risks* with this proposal. For each risk, identify one associated *uncertainty* and one associated *measurement* that might be taken to reduce that uncertainty.

(c) *(4 pts.)* Support or refute the claim that a "passaround *triage* review" process would be have a net benefit compared to a current practice in which initial triage involves a single decision maker. You can define passaround triage review as you like, but it should feature an initial triage proposal being evaluated and commented on by other developers, analogous to code review.

(d) *(3 pts.)* Support or refute the claim that it should be simple and quick for any software developer at your company to build and test any software in your company's repository. Use two examples as evidence.

# 4 Mutation Testing (16 points)

Consider this method to compute the factorial of a number. The original program is shown on the left; three first-order mutants are each indicated by a comment on the right.

```
1  def factorial(n):
2    i = 1
3    fact = 1
4    if (n < 0):              # Mutant 1 has n <= 0
5      return "error"
6    else:
7      while (i <= n):        # Mutant 2 has i < n
8        fact = fact * i      # Mutant 3 has fact = fact + i
9        i = i + 1
10     return fact
```

(a) *(12 pts.)* Complete the table below by indicating whether or not each test kills each Mutant. Write "K" for Killed and "N" for not killed.

| | Input (n) | Oracle | Mutant 1 | Mutant 2 | Mutant 3 |
|---|---|---|---|---|---|
| Test 0 | 0 | 1 | | | |
| Test 1 | 1 | 1 | | | |
| Test 2 | 2 | 2 | | | |
| Test 3 | 3 | 6 | | | |

(b) *(1 pt.)* What is the mutation score for Tests 0–3 using Mutants 1–3?

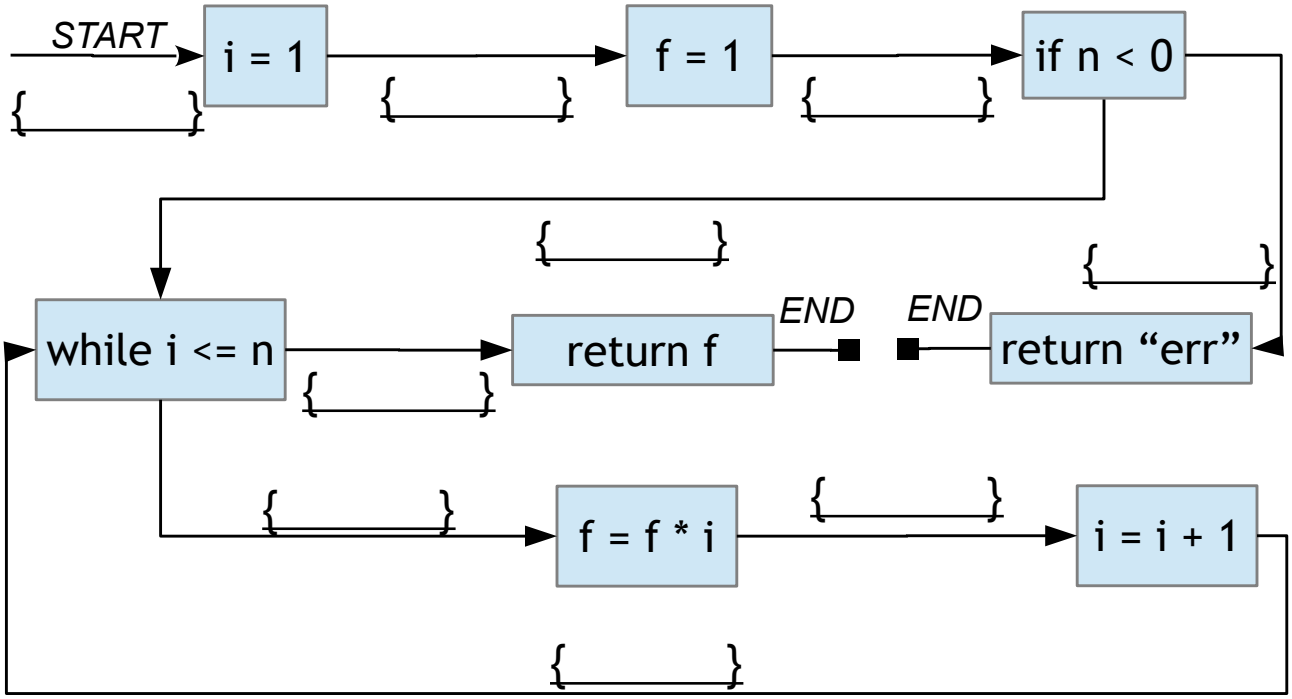
(c) *(1 pt.)* What is the mutation score for only Tests 0–**1** using Mutants 1–3?

(d) *(2 pt.)* Support or refute the claim that a *higher-order* mutant that combines Mutation 1 and Mutation 3 is useful.

# 5 Dataflow Analysis (20 points)

Consider a *live variable* dataflow analysis for *three* variables, `i`, `f` and `n`. We associate 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 `i` and `f` are alive but `n` is not, we write { `i, f` }. The special statement `return` reads, but does not write, its argument. (You must determine if this is a forward or backward analysis.)

*(18 pts.)* Complete this live variable dataflow analysis for `i`, `f` and `n` by filling in each blank *set* of live variables.




*(2 pts.)* Support or refute the claim that a live variable dataflow analysis always terminates, even on programs that contain loops.

# 6 Quality Assurance Analyses (14 points)

(a) *(3 pts.)* Describe a situation under which the *Eraser* lock-set analysis might report a false positive race condition on a shared variable.

(b) *(4 pts.)* Support or refute the claim that the task of finding race conditions is more suited to dynamic analyses than to static analyses.

(c) *(3 pts.)* Support or refute the claim that information in a bug report is more likely to help guide an automated static analysis (rather than an automated dynamic one).

(d) *(4 pts.)* Give three advantages of *abstraction* in analysis and one disadvantage of it.

# 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?

*(Feedback)* What is one thing you would change about the department or the major?

*(My Choice Psych)* What's the difference between the *backfire effect* and *confirmation bias*?

*(My Choice Reading)* In Petrović and Ivanković's *State of Mutation Testing at Google*, what did they do to make it easier to understand the results of the analysis?

*(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!).