EECS 481 — Software Engineering
Spring 2020 — Exam #1

- There are seven (7) questions in this exam, each with multiple parts. Some questions span multiple pages. If you get stuck on a question, move on and come back to it later.
- Once you download this exam, you have two (2) hours to complete and upload it. If you encounter technical difficulties, email the staff immediately.
- This exam is open book, notes, and Internet. *You may not communicate with others while completing this exam.* You can email the staff, make *private* Piazza posts, or use Slack to send direct messages to staff. We will try to respond during the hours of 11AM to 11PM Eastern time on Friday, Saturday, and Sunday.
- You will complete the exam by filling in the accompanying `exam-answers.txt` files. Once complete, submit `exam-answers.txt` alone to the course website: `https://dijkstra.eecs.umich.edu/kleach/eecs481/shibboleth/exam-submit.php`.
- 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.
- 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: Generated for kjleach.

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 *exactly one* corresponding answer.) Each option *can* be used more than once.

A. Alpha Testing	B. Agile Development	C. Beta Testing	D. Build Automation
E. Comparator	F. Dataflow Analysis	G. Development Process	H. Dynamic Analysis
I. Formal Code Inspection	J. Invariant	K. Integration Testing	L. Maintainability
M. Mocking	N. Oracle	O. Passaround Code Review	P. Path Predicate
Q. Perverse Incentive	R. Quality Property	S. Risk	T. Requirements
V. Software Metric	W. Spiral Development	X. Static Analysis	Y. Threat to Validity
Z. Waterfall Model			

(a) H A developer decides to use `gdb` to debug a segmentation fault at runtime.

(b) B The manager of the *Sprightly Software Company* decides to plan effort in two-week sprints, with daily stand-up meetings.

(c) Q The leadership of a software company decides to provide bonuses to developers based on the average number of bugs fixed per 1,000 lines of code.

(d) S In class, we discussed pacemaker software that allowed attackers to kill patients by wirelessly disabling pacemakers. A company responsible for such software can considering adopting various processes to help account for this aspect of project delivery.

(e) W The database company *Debacle* decides to adopt a procss where they deliver prototypes every three months. Each prototype phase, they gather any changing requirements, design and implement software that meet these new requirements, and deliver the next prototype.

(f) P While inspecting a program with many sequential if statements, you carefully design a test input that causes a specific set of if statements to be taken.

(g) Y In an effort to improve the readability of your code, your teammate reduces its Halstead volume. He says that maintainable code must have a minimal Halstead volume. However, you point out that his most recent commit was an unreadable mess with a deceptively low Halstead volume. What does that say about the applicability of the Halstead volume?

(h) V Visual Studio embeds a numerical computation for approximating the complexity of code.

(i) I The company *Repo Men* writes software for prosthetic hearts. They should consider adopting a process for thoroughly discovering defects. One manager proposes gathering a team of five engineers to prepare comments and evaluate checklists with respect to source code.

(j) J A function `Circumference` accepts a single floating point parameter, `radius`, always returns the value 2*PI*`radius`.

(k) K In a certain development team, both the user interface and database connector are complete and pass all test cases. However, a new bug seems to pop up when the software runs end-to-end.

(l) G In an effort to modernize, the company *Pineapple* decides to release prototypes on a more frequent basis. Additionally, engineers must use 2-space Tabs and adopt the use of CamelCase in their code.

(m) C After gathering feedback from other developers, the *Happy Lizards* game team allows users to pay for an early release of software to gather feedback.

2 Testing and Coverage (22 points)

Consider the following program. Statements of interest are labeled S_1 through S_5.

```
1  int amazing (int x, int y, int z) {
2    if ( x > 0 ) {
3      S_1;
4      if ( z < 3 ) {
5        S_2;
6      } else {
7        S_3;
8      }
9    }
10   if ( x != -1  ||  y == 0  &&  z > (2 * y) )  {
11     S_4;
12   } else {
13     S_5;
14   }
15 }
```

(a) *(1 pt. each row)* In the table below, identify integer values for x, y, and z that result in the coverage specified in the table, or indicate that it is not possible.

Statements covered	x	y	z	Not possible
S_2, S_4				N
S_4	-6	-6	-6	
S_2, S_3, S_4				N
S_2, S_3, S_4				N
S_1, S_2, S_5				N
S_1, S_4				N
S_1, S_2, S_4	1	-6	-6	
S_1, S_3, S_5				N

Answers will vary. The bank of statements often required conflicting, impossible predicates. For example, if you see S1, then you *must* cover S3 or S4 as well.

Next, consider the function below. Against, statements of interest are labelled S_1 through S_4.

```
1  int ShuaiDaiLe (int x, int y, int z) {
2    if ( x != 1 ) {
3      S_1;
4      if ( y > 3  &&  z < 0 ) {
5        S_2;
6      } else {
7        S_3;
8      }
9    }
10   if ( x > 1  &&  y == x )  {
11     S_4;
12   }
13 }
```

(b) *(2 pts. each row)* In the table below, you are given values of x, y, and z. In the corresponding blanks, fill in the Path Coverage *count* for each test case (i.e., report how many unique paths are executed; do *not* report a percentage).

x	y	z	Path coverage (count)
0	0	1	1
2	1	0	1
-1	-1	-1	1
0	1	2	1
0	-1	-1	1
-1	-1	2	1

All answers were 1. Every set of variables executes exactly one path. (An initial revision of this question was much more involved; unfortunately, later revisions mistakenly oversimplified this problem)

(c) *(2 pts.)* Identify any one path predicate from the program above that executes a unique path through the program.

3 Short Answer (20 points)

(a) (1 pt.) In a few words, identify which phase of software development costs the most amount of money and resources?

Maintenance. Lecture and readings covered industrial reports about costs of fixing defects at various phases of development. Maintenance rules all.

(b) (1 pt.) In one sentence, describe what the `afl-gcc` program does to source code.

afl-gcc is a special gcc compiler for the AFL tool suite that instruments source code to help determine path coverage as part of AFL's fuzzing methodology.

No partial credit. Answer must include instrumentation and coverage.

(c) (2 pts.) In two or fewer sentences, support or refute the claim that all developers should participate in alpha testing at the end of every sprint or project milestone.

Support: Sufficient testing will increase delivered software quality and keep developers on track. This is particularly relevant in safety-critical software, where alpha testing may reveal severe defects.

Refute: The resources required to implement alpha testing every sprint may detract too much from productive development. The benefit of the testing may not make up for the decreased productivity.

Other answers may apply. No partial credit.

(d) (3 pts.) Identify *two* risks associated with adopting dynamic analysis techniques at a company that currently does not use any. Identify a *measurement* that could be used to reduce each risk.

Instrumentation for DA may incur runtime performance penalty. Need to measure execution time and slowdown caused.

Developer effort required to bootstrap a new DA (e.g., a new a test suite), perhaps resulting in delays in current plans or projections. Company could fold DA development into current process; start measuring developer activity wrt new DA.

DA rig may introduce complexity in the repository. What if DA takes too long? Start measuring time taken to complete DA.

DAs introduce statistical errors. If we replace sound SA with unsound and incomplete DA, developers may miss issues that would have been caught by SA. Measure types of defects fixed resulting from DA vs. SA.

Others may be correct. Evaluated case by case. -1 for each risk or measurement not present, down to a minimum of 0/3.

(e) (3 pts.) A fresh startup called *FaceBack* has designed software that can identify the back of a person's head given a picture of their face. Some team members support adopting an official company list restricting which programming languages can be used as the team grows. In three or fewer sentences, support or refute the use of such approved programming language lists in terms of risk management and process.

Support: A restricted set of programming languages may ease code reviews as reviewers know what to expect (lowered risk, process simplification). It will also lower overhead associated with enforcing code style (lowered risk). There may also be benefits to simplifying integration testing (process simpliciation).

Refute: May diminish developer productivity, especially if parts of the team are more familiar with one language over another (increased risk and uncertainty; how to plan/allocate developer effort?). Additionally, such a practice may increase defects among developers who are not familiar with approved languages (increased risk). Encourages implementation bias by eliminating potential languages that may simplify some aspect of the project (increased risk, process complexity).

Other answers may apply. Partial credit given if discussion of risk or process is present, but not both.

(f) (2 pts.) Consider a program with five sequential `if` statements that accepts five boolean inputs. Assuming each condition evaluates a single unique input, what is the minimum number of test cases required to achieve 100% Path coverage? Condition coverage?

Sequential if statements means that there are 32 possible paths to cover. The assumption is that each of the 5 if's has one input to evaluate, so condition coverage is achieved with 10 tests (2 for each outcome of each if).

No partial credit.

(g) (3 pts.) In three or fewer sentences, support or refute the claim that Microsoft's Maintainability Index helps identify difficult-to-read code.

Support: As a proxy for code size, in the limit, the Index can help identify grossly unmaintainable code. As long as it is applied judiciously, it can help make first order decisions about whether code should be revisited for maintainability.

Refute: As a proxy for code size, the Index does not help you identify subtle differences in maintainability, especially across different code. One method may be perfectly readable, but have a low Index, while another method may be disastrous in comparison, yet have a higher Index.

Other answers may apply. Partial credit if answer does not speak to subtlety of applying this metric to compare code.

(h) (1 pt.) Give one example of a tool used for code review.

GitLab. There were other examples shown in lecture, such as an integrated Visual Studio tool.

(i) (2 pts.) Suppose you are building a large C program that comes with a `configure` script. Further suppose that you want to use a custom version of `gcc` located in `/usr/local/bin/481-gcc` and pass the flags `-O3 -Wall`. Assuming you are working in the same directory as the `configure` script. Identify the command input to successfully configure this project.

./configure CC=/usr/local/bin/481-gcc CFLAGS="-O3 -Wall"

(j) (2 pts.) In three sentences or fewer, describe the differences between spiral development and waterfall development.

Full credit answer will discuss how spiral relies on continuous releases of prototypes to reduce risk. Waterfall is divided into discrete phases over the course of an entire project. Spiral contains aspects of waterfall, just iterated multiple times during a project.

4 Invariant Detection and Mutants (15 points)

Consider the code snippet below:

```
1  int rose ( int a, int b ) {
2      int x = 0;
3
4      if (a <= 0 || b <= 0 ) {
5          return -1;
6      }
7      if (a > 6 || b > 6) {
8          return -1;
9      }
10
11     if (a == 3 || a == 5) {
12         x += (a - 1);
13     }
14     if (b == 3 || b == 5) {
15         x += (b - 1);
16     }
17     return x;
18 }
```

(a) *(3 pts. each row)* In the table below, several candidate invariants are listed. For each candidate, EITHER (1) specify a test case in terms of a and b that *falsifies* the candidate, OR (2) identify a first-order mutant that leads you to *retain* the candidate. (Fill in the appropriate column).

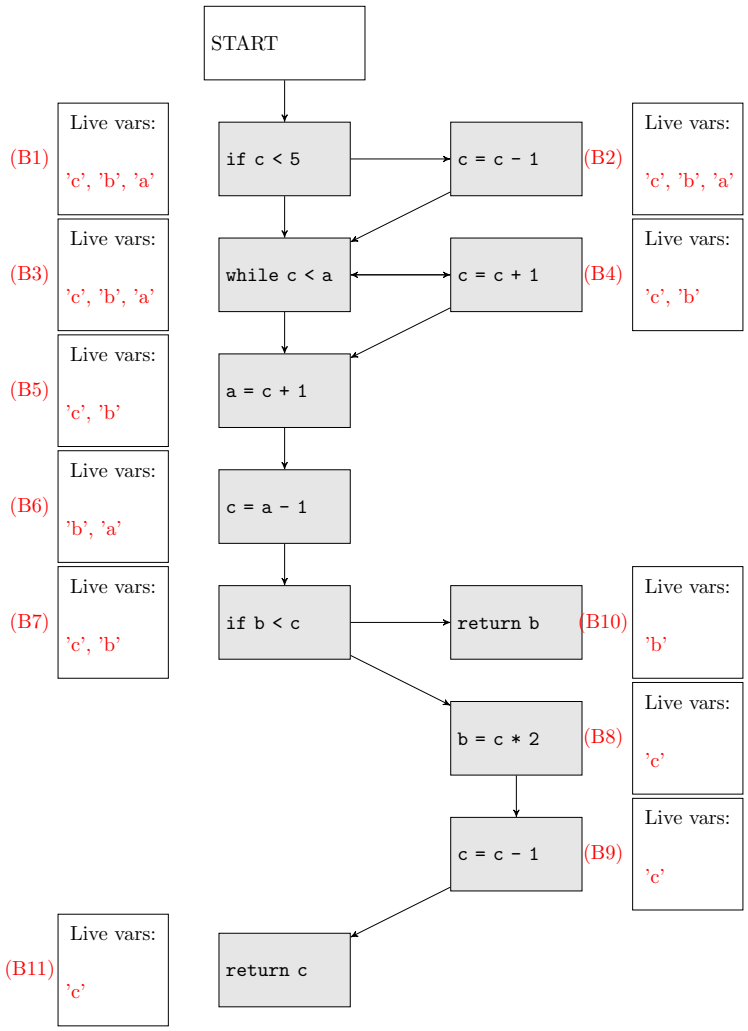
Invariant Description	a	b	Falsifying mutation (if needed)
rose <= 2b for all a and b	a=5	b=1	
rose is nonnegative and even for all a and b	a=0	b=0	
rose is even for all a and b	a=0	b=0	
rose <= a + b for odd a and b			Change (a-1) to (a+1)
rose < 12 for all a and b			Change return x to return x*2

Answers can vary. *One banked invariant was impossible to answer, thus everyone receives a minimum of 3 points on this question

5 Dataflow Analysis (20 points)

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

(a) *(18 pts.)* Complete this live variable dataflow analysis for `a`, `b` and `c` by filling in each Live Vars. box with the *set* of live variables *just before* that point in the program.



(b) *(2 pts.)* Support or refute the claim that a statement with no live variables can be removed without affecting the program's correctness.

Support: Statements with dead variables implies the statement affects nothing. Such dead code can be removed without loss. **Refute**: Statements like $x = 1/0$ might appear dead, but removing them may influence intended behavior.

6 Dynamic Analyses (10 points)

Consider a concurrent system in which two threads share a common variable, `m`. Each thread is executing the code shown below.

```
1  while (true) {
2    lock(A);
3     m := 6;
4     m := m + 1;
5    unlock(A);
6    m := 8;
7    lock(B);
8     m := m + 1;
9    unlock(B);
10 }
```

```
1  while (true) {
2    lock(A);
3     m := m + 1;
4    unlock(A);
5    lock(B);
6     m := m + 1;
7    unlock(B);
8  }
```

Assume that two mutual exclusion locks are present: A and B. In this setup, the `lock` method acquires the named lock, while `unlock` releases it. If thread 1 attempts to run `lock(A)` but thread 2 currently holds the lock, then thread 1 must wait until thread 2 runs `unlock(A)`.

(a) *(2 pts.)* Identify whether any race conditions are present with respect to variable m.

Answers will vary. Both locks A and B must be acquired to safely modify variable m. If one thread acquires lock A but not B, then there is a race condition.

Note that we technically are not talking about deadlock here, which is a separate issue. It is possible for a circular wait to occur. Thread 1 acquires A, thread 2 acquires B, Thread 1 attempts to acquire B, thread 2 attempts to acquire A. Deadlock.

(b) *(8 pts.)* Justify your answer. *If there is a race condition present*, specify the line number, then explain the sequence of events that exposes the race condition. *If there is no race condition*, explain why the calls to `lock` and `unlock` will never allow unsafe access to variable m. In either case, you can say things like "Thread 1 acquires lock A" or "Thread 2 writes $x + 1$ to variable y."

Answers can vary. You will get credit if you indicate operations leading to a race condition, or if you indicate that both threads acquire both locks before modifying m, depending on which is appropriate for your version of the question.

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

(a) *(Feedback)* What is your least favorite thing about this class?

Any non-blank answer receives credit. (correct answer: NOTHING)

(b) *(Feedback)* What is your most favorite thing about this class?

any non-blank answer receives credit. (correct answer: EVERYTHING)

(c) *(Pyschology)* Provide an example of confirmation bias.

Any example of confirmation bias will receive credit. Example: I believe the Earth is flat. I go to Kansas and see a lot of flat land, confirming my belief.

(d) *(Psychology)* Explain the McNamara fallacy in your own words.

Strictly using quantitative metrics for decision making, ignoring other factors. For example, students may believe that mastering an autograder rubric means that they have mastered all class material.

(e) *(Random)* What is your favorite text editor?

We used this to look for discrepancies from the Quiz question. The correct answer is vim!

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

We're tough on these. Basically, we're looking for something more than copying and pasting text from an optional reading. For example, "it seems like CHESS is quite powerful and can be used to rigorously debug concurrent programs by treating scheduler interleavings as input."

(g) *(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!).