1. (1pt) **Name:** _____

INSTRUCTIONS: Carefully read each question, and write the answer in the space provided. If answers to free response questions are written obscurely, zero credit will be awarded. The correct answer to a free response question will never contain any significant words used in the question itself (i.e., "crossword rules"). You are permitted one 8.5x11 inch sheet of paper (double-sided) containing notes; all other aids (other than your brain) are forbidden. Questions may be brought to the instructor.

For **TRUE** or **FALSE** and multiple choice questions, circle your answer.

On free response questions only, you will receive **20%** credit for any question which you leave blank (i.e., do not attempt to answer). Do not waste your time or mine by making up an answer if you do not know. (Note though that most questions offer partial credit, so if you know part of the answer, it is almost always better to write something rather than nothing.)

To get credit for this question, you must:

- Print your name (e.g., "Martin Kellogg") in the space provided on this page.
- Print your UCID (e.g., "mjk76") in the space at the top of **each** page of the exam.

Contents (blanks for graders only):

Writing your name on every page:	1 / 1
I. Reading Quiz Redux:	5 / 5
II. Very Short Answer:	29 / 29
III. Short Answer:	37 / 37
IV. DBQs:	28 / 28
V. Extra Credit:	6 / 0
Total:	106 / 100

I. Reading Quiz Redux (5pts)

2. (1pt) **Tech Debt, Part 1: TRUE** or **FALSE**: all technical debt is the result of programmer laziness.

3. (1pt) **Requirements, Part 2:** The author describes formal specifications as providing three main benefits. Which of the following is NOT one of those:

   A It provides clear documentation of the system requirements, behavior, and properties.

   B It clarifies your understanding of the system.

   C It finds really subtle, dangerous bugs.

   D It makes writing the code quicker and easier.

4. (1pt) **Free and Open Source Software:** The author claims that the term "free software" means:

   A software you can get for zero price

   B software which gives the user certain freedoms

   C software whose source code you can look at

   D none of the above

5. (1pt) **Static Analysis, Part 1:** FindBugs _____:

   A always warns about line X if it is possible there is a bug on line X

   B never warns about line X unless there is definitely a bug on line X

   C both A and B

   D neither A nor B

6. (1pt) **DevOps, Part 2:** Which of the following does Dan Luu advocate for when making a high-risk change?

   A having multiple people watch or confirm the operation

   B having ops people standing by in case of disaster

   C automating the change instead of letting a human do it

II. Multiple Choice and Very Short Answer (29pts). In the following section, either circle your answer (possible answers appear in **bold**) or write a very short (one word or one phrase) answer in the space provided.

7. (2pt) Google (and other big tech companies) design their hiring process to avoid false **positive** / **negative** results: that is, to avoid hiring unqualified candidates, even if some good candidates are rejected.

8. (2pt) A **functional specification** / **quality requirement** is a description of what a system should do that doesn't specify how the system should do it.

9. (2pt) Alice's deadline to deliver a feature is today. She finishes writing the feature and tests it on her local machine, but chooses not to write automated tests, even though she knows that it is risky, because she wants to meet her deadline. This is an example of **technical debt**

10. (2pt) Which of the following is NOT a static analysis:
    A dataflow analysis
    B testing
    C code review

11. (2pt) Which of the following is it best practice to commit to your version control system? Circle all that apply.
    A credentials
    B code
    C binary files
    D config files

12. (2pt) **TRUE** or **FALSE**: continuous integration can only be used by DevOps teams

13. (2pt) A benefit of modern code review is that more than one person has seen each piece of code that is checked in. This benefit reduces your team's **bus factor; "surprise" got -1**

14. (2pt) A tenet of the **Waterfall** / **Agile** methodology is to always have a working prototype.

15. (2pt) **TRUE** / **FALSE**: modern code review is based on an inductive argument for the quality of software: if the previous version is good, and the change is good, then modern code review relies on the composition of the previous version and the change being good.

16. (2pt) You are a manager at MTa (pronounced "meta"), a social media company for NYC Subway aficionados. One of your employees would like to modify an open-source library that is licensed under the GNU Public License, version 2 (GPL v2). You veto this decision, because the GPL v2 is a **copyleft or "free software"** license: it would force you to release your modifications in the open, too.

17. (3pt) In a single component with a model-view-controller architecture, how many of each of the following should there be?

Models:	**0-1**	**exactly 1**	**1 or more**	**at least 2**
Views:	**0-1**	**exactly 1**	**1 or more**	**at least 2**
Controllers:	**0-1**	**exactly 1**	**1 or more**	**at least 2**

18. (2pt) **TRUE** or **FALSE**: because we have the singleton design pattern, global state is a good design choice

19. (2pt) You are writing a compiler for your new language, Java++ (it's like Java, but with more classes!). A user reports that a particular program doesn't compile correctly. You add that program to your test suite, and then you start fixing the problem in your compiler. You are practicing
test-driven development; half-credit for "regression testing"

20. Consider a program with seven sequential if statements that accepts seven boolean inputs. Assuming each condition evaluates a single unique input, what is the minimum number of test cases required to achieve:

    (a) (1pt) 100% branch coverage? **2: one all false, one all true**

    (b) (1pt) 100% condition coverage? **14 = 2*7**

III. Short answer (37pts). Answer the questions in this section in at most two sentences.

21. For each sub-problem, specify if delta debugging can be used to solve the problem. If it can, provide a brief description of an "Interesting" function that will help solve the problem. If it cannot, specify which properties of delta debugging make it not suitable to solve the problem.

    (a) (3pt) Given a list of (positive and negative) integers that sums to zero, we want to find the minimal subset that sums to zero.
    Delta debugging is not suitable for this use case: integer summation is ambiguous.

    (b) (3pt) In an effort to reduce the memory footprint of a Java program, we decide to try replacing every `long` variable with an `int`. However, when we do so, our program fails some of its test cases (we suspect due to integer overflow). We want to identify which `long` variables can be replaced with `int` variables such that the program still passes all tests.
    Delta debugging is suitable for this use case. We can define script `is_interesting.sh` such that it takes a list of occurrences of long and replaces each with an int. The script exits 1 if the code compiles and runs the tests successfully, and it exits 0 if the code doesn't compile or fails any tests.

    (c) (3pt) In an effort to reduce the build time of a Java program, we decide to try removing dependencies from our build file: the program is old, and we believe some of the dependencies are no longer used. However, when we do so, our program fails to compile (we suspect because some of the dependencies we removed are still in use). We want to identify the minimal subset of dependencies that enables our program to compile.
    Delta debugging is suitable for this use case. We can define script `is_interesting.sh` such that it takes a list of dependencies, removes them from the build file, and then attempts to compile the program. The script exits 1 if the code compiles successfully, and 0 if it does not. Also "no" is acceptable if and only if the justification is "dependencies may depend on each other, so the problem may be non-monotonic". '

22. (4pt) Write a method accepting one input parameter for which test input generation via constraint solving will work better than test input generation at random.
    Answers vary. The key idea is to include a very specific test that only a single input can satisfy, such as `if (x == 10): ...`. Common mistakes included: describing a program but not giving code/pseudocode (-1); giving a very complex program that would be hard for both kinds of testing techniques (-2), or giving a 50/50 if statement, like `if (x > 0)` (-3).

23. For each of the following bugs, describe a situation in which the bug would be *high severity* and describe another, distinct situation in which the bug would be *low severity*.

    (a) A programmer-managed resource, such as a socket or database connection, is not closed before the last pointer to it goes out of scope.

        i. (1pt) High severity: **A long-running program, such as a webserver, which may eventually run out of resources; another common answer was a database holding sensitive info, which might be vulnerable to attack.**

        ii. (1pt) Low severity: **a short-running program, such as one that sends a single ping and then exits: the resource will be freed when the program ends, anyway. Other examples are possible. The most common mistake was "webserver that holds unimportant info", which can still sufffer resource starvation due to a resource leak.**

    (b) An integer value is stored in the wrong units (e.g., feet vs meters, seconds vs hours, etc.).

        i. (1pt) High severity: **Mars polar orbiter crash. Other answers are possible.**

        ii. (1pt) Low severity: **The integer value is part of a game and is only displayed to the user: e.g., it doesn't matter if an imaginary car's speed is in m/s or mph. Other answers are possible.**

24. Consider the following pairs of tools, techniques, or processes. For each pair, give a class of defects or a situation for which the first element performs better than the second (i.e., is more likely to succeed and reduce software engineering effort and/or improve software engineering outcomes) and explain why.

    (a) (3pt) DevOps approach to operations better than traditional ("sysadmin") approach to operations
    DevOps is best when the organization developing a service is also running that service, becasue operational pain is felt by the same org. A common mistake here was just defining DevOps rather than giving a scenario or situation.

    (b) (3pt) Watchpoints better than breakpoints
    Watchpoints are better when you know what value shouldn't be changing, but not where in the code that change is occuring.

    (c) (3pt) Modern code review better than integration testing
    The easiest answer is that code review can find code-design defects, but integration testing cannot. Many other answers are possible.

25. (7pt) Consider a static analysis that computes, for every program point, which variables contain a value that has definitely been used by the program up to that point. (This is a form of *strictness analysis*.) The analysis associates an analysis fact with each variable at each program point: either "T", meaning "it is not known whether the variable has been used yet" (aka "top"); or "U", meaning "the variable has definitely been used at this point." The analysis has a transfer function for binary addition: after the statement "x = y + z", "y" and "z" are both "U". It also has a transfer function for "return x": after such a statment, "x" is "U". Other transfer functions are standard for a forwards, "definitely"-style analysis (like the nullness analysis described in class). Simulate this analysis on the following program, filling in each blank with the corresponding abstract value ("T" or "U") that the analysis would compute for that variable at that point. (The "*" boolean operator means "choose at random", or equivalently, "flip a coin".)

```
1  def f(a, b, c) {
2                     [ a -> T , b -> T , c -> T , d -> T  ]
3    if (*) {
4                     [ a -> T , b -> T , c -> T , d -> T  ]
5      d = a + b;
6                     [ a -> U , b -> U , c -> T , d -> T  ]
7    } else {
8                     [ a -> T , b -> T , c -> T , d -> T  ]
9      d = a + c;
10                    [ a -> U , b -> T , c -> U , d -> T  ]
11   }
12                    [ a -> U , b -> T , c -> T , d -> T  ]
13   return d;
14                    [ a -> U , b -> T , c -> T , d -> U  ]
15 }
```

26. (4pt) You are a software engineer at Orange, a computer hardware company with a software division. You are responsible for replacing a microservice that determines how much money customers owe to various app store vendors at the end of each month; your service sends this information to another service that actually bills the customer. A legacy service exists but takes all month to run. Your boss wants you to rewrite it to improve its performance, so that you can compute the customer's bill at any time, not just once a month. However, your boss is also concerned about correctness. Describe a low-cost strategy that you could apply in this situation to ensure that your new service has the same behavior as the slow, legacy version. Answer in at most 3 sentences.

The correct thing to do here is to apply differential testing (aka "tests for free"): log all traffic to the old service, and check that the new service produces the same answers as the old one for each input for some period of time (until your boss is satisfied, probably).

IV. Document-based Questions (28pts). All questions in this section refer to a documents **A-B**. These documents appear at the end of the exam (I recommend that you tear them out and refer to them as you answer the questions).

Questions on this page refer to **Document A**. Assume that the only statements we are interested in are `STATEMENT 1` through `STATEMENT 4`.

27. (2pt) Write a test suite for `foo` with 50% statement coverage. Express your answer as a list of tuples, e.g., "(x = #, y = #, z = #), (x = #, y = #, z = #), etc.", where each "#" is a specific integer.
(x = 6, y = 0, z = 3)

28. (2pt) Is it possible to achieve 100% statement coverage on this method? If so, provide a test suite that has 100% statement coverage. If not, why not?
It is not possible, because STATEMENT 2 is unreachable.

29. Consider the mutation operator "< to <=".

    (a) (2pt) How many first-order ("mutated only once") mutants can be produced by applying this mutation operator to `foo`?
    2

    (b) (2pt) What is the mutation score of a test suite containing only the following test input (assume that executing a different statement causes the mutant to be killed): (x = 5, y = 2, z = 5)?
    0%

    (c) (3pt) Is it possible to change one of x, y, or z in the test input in the previous part of the question to improve the mutation score? If so, state which of x, y, and z to change, what its new value should be, and what the new mutation score is. If not, explain why not.
    z = 6 changes the mutation score to 50%

    (d) (3pt) Is it possible to change one of x, y, or z in the test input in the previous part of the question to improve the mutation score again? If so, state which of x, y, and z to change, what its new value should be, and what the new mutation score is. If not, explain why not.
    It is not, because the second mutant is an equivalent mutant.

Questions on this page refer to **Document B**.

Suppose that you are a software engineer at `codewith.us`, a non-profit that provides computer science education services to K-12 students via an online coding platform. Your boss sends you the email in **Document B**.

30. (14pt) Write a response to your boss. Your response must either provide a reasonable, well-justified plan for how you will implement the proposed feature (including task sizing and at least research, implementation, and testing tasks) or a well-reasoned justification for why the proposed feature is not feasible.

The proposed feature is an instance of the halting problem, and Mr. Rice's insistence that you get the correct answer 100% of the time means that it is impossible to implement the feature as described. Correct answers must be phrased as an email, and use a polite but firm tone. Common mistakes included:

- **not identifying that this was the halting problem at all, and saying it would be easy (-10)**
- **not connecting Mr. Rice's requirement to the difficulty of the task ("the halting problem is impossible", -4)**
- **giving a rough explanation of the halting problem, but not directly identifying it (-1)**
- **failing to offer any alternatives after saying that the halting problem is impossible (-4 for no alternatives, -2 if only one alternative was offered without giving choices to Mr. Rice)**
- **vagueness (deduction varies)**

V. Extra Credit. Questions in this section do not count towards the denominator of the exam score.

31. (1pt) Why did panelist Rupali Vohra (Convoy) say that her undergraduate operating systems class was particularly useful to her in her software engineering career? **Any answer involving the common-ness of distributed systems or the similarity between multithreading and distributed systems is acceptable.**

32. (1pt) The "**campsite**" approach to managing technical debt, according to our panelists, involves leaving any code you modify in less technical debt than your found it in.

33. (1pt) Name an optional reading assignment that you read but will not and have not used as a response to the "Optional Reading Response #1" or "Optional Reading Response #2" assignments on Canvas, and then give a one-sentence description of something you learned from that reading. (Note: answering this question with a particular reading means you cannot use that reading for your "Response #2", due on May 2.) **Answers vary.**

34. (1pt) Name another optional reading assignment, distinct from your answer to question 33, that you read but will not and have not used as a response to the "Optional Reading Response #1" or "Optional Reading Response #2" assignments on Canvas, and then give a one-sentence description of something you learned from that reading. (Note: answering this question with a particular reading means you cannot use that reading for your "Response #2", due on May 2.) **Answers vary.**

35. (1pt) Name a recent "hot topic" in Software Engineering research and give a one-sentence description of that topic. **Expected answers are the topics covered in class on 20 April, but any topic on which there were 3 or more papers at FSE, ICSE, ISSTA, or ASE in the past 2 years is acceptable.**

36. (1pt) Would you be willing to serve on an engineer panel in the future (i.e., once you have a job)? If so, leave an email address that you expect to still monitor after graduating in this space. If not, write "no" to receive full credit.**"no" or any email address receives full credit.**

Document A:

```
1   void foo(int x, int y, int z) {
2     if (x > 5) {
3       // STATEMENT 1
4     else if (y < 4 && x > 7) {
5       // STATEMENT 2
6     }
7     if (z < 6) {
8       // STATEMENT 3
9     } else {
10      // STATEMENT 4
11    }
12  }
```

Document B:

[Your Name],

I met with the executive board today, and they're onboard with a new feature to help our students avoid writing programs that run forever by warning them before they start a program whether it will loop forever. I'd like you to design the feature and then write me up an estimate for how much time it'll take you to implement. Here are some requirements I gathered from the board:

- All the board members agreed that the warning message needs to be a bright red popup that appears when the student is about to run a program that'll run forever.
- Ms. Wheat (she used to be a teacher, remember, so she really gets the students!) insisted that we use only simple words in the warning message. I think her suggestion was "Watch out! This might last forever!", but we can iterate on that.
- Mr. Rice (who, as you might remember, is our biggest donor - so we have to keep him happy) insisted that if we're going to spend technical resources on this and not on his pet project of making all of our buttons larger, we really have to get it right every time.
- All the board members agreed that you should use a font that really pops out and makes the kids rethink what they were about to do. They all still think it's the 90s, though, so we can probably not worry about this too much as long as the design you pick gels with the rest of the site.

Get back to me ASAP with that estimate - I want you working on this next sprint.

Thanks! [Your Boss]