

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 with a short answer (i.e., one word or phrase) 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 **hand-written** 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.

	Writing your UCID on every page:	<u>1</u> / 1
	I. Reading Quiz Redux:	<u>3</u> / 3
	II. Very Short Answer:	<u>19</u> / 19
	III. Matching:	<u>20</u> / 20
Contents (blanks for graders only):	IV. Short Answer:	<u>31</u> / 31
	V. “Your Choice” Reading Quiz	<u>2</u> / 2
	VI. DBQs:	<u>24</u> / 24
	VII. Extra Credit:	<u>8</u> / 0
	Total:	<u>108</u> / 100

I. Reading Quiz Redux (3pts)

2. (1pt) **Code Review:** today's reading was an engineering practices guide associated with which of the following companies:
- A Microsoft
 - B Amazon
 - C Netflix
 - D none of the above
3. (1pt) **Static Analysis, Part 1:** How many "infinite recursive loop" bugs did FindBugs find in Google's codebase?
- A 0
 - B 1
 - C more than 70
4. (1pt) **Testing, Part 3: TRUE or FALSE:** A well-written C program will typically contain some defensive conditionals which in practice are always true or always false. This leads to a programming dilemma: does SQLite remove defensive code in order to obtain 100% branch coverage?

II. Multiple Choice and Very Short Answer (19pts). 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.

5. (1pt) A **static** / **dynamic** analysis involves running the program being analyzed.
6. (2pt) The Agile manifesto includes which of the following principles? Select all that apply.
- A process and tools over individuals and interactions
 - B working software over comprehensive documentation
 - C contract negotiation over customer collaboration
 - D responding to change over following a plan
7. (1pt) **TRUE** or **FALSE:** you should follow the established conventions of a codebase when modifying that codebase, even if those conventions violate the usual principles of code-level design.
8. (2pt) When choosing test inputs by hand, it is good practice to include **edge cases** like 0, null, or empty lists or files.
9. (2pt) Which of the following are best practices related to continuous integration (CI)? Circle all that apply.
- A developers don't need to run tests locally at all, since all tests will run in CI
 - B the CI build should not be permitted to fail for long periods of time
 - C developers should run the full CI build on their machine before committing
 - D all changes to the project should be gated by CI

10. (2pt) Which of the following are build systems useful for automating? Circle all that apply.
- A** installing dependencies
 - B** compiling the code
 - C** creating artifacts for customers
 - D** running tests
11. (2pt) Debugging **flaky** tests is unusually difficult, because they sometimes fail and sometimes pass in a seemingly-non-deterministic manner.
12. (1pt) **TRUE** or **FALSE**: “1/3 finished with implementation” is a reasonable milestone to include in a planning document like your project proposals.
13. (2pt) When a version control system like `git` determines that two changes are conflict-free, which of the following are possible for the merged code? Circle all that apply.
- A** compilation errors
 - B** multiple changes to the same line
 - C** test failures
14. (2pt) A common pathology for poor-performing software engineering teams is a non-**hermetic** build process: for example, because there are dependencies in the build on projects that are checked out locally on the computer of each member of the development team.
15. (2pt) Which of the following program analyses are sound? Circle all that apply.
- A** code review
 - B** a static analysis that reports a bug on every line
 - C** static type systems, like those in Java or Rust
 - D** a static analysis that never reports a bug

Table 1: **Answer Bank:**

A. Mutation Analysis	B. Functional Specification	C. Implicit Oracle
D. Halting Problem	E. Feature Branch Development	F. Project Manager (PM)
G. Linting	H. Dataflow Analysis	I. Test-driven Development
J. Centralized Version Control	K. Functional Programming	L. Standup
M. Sprint	N. Control-flow Graph	O. Explicit Oracle
P. Behavioral Interview	Q. Magic Number	R. Hermeticity
S. Distributed Version Control	T. Pair Programming	U. Modern Code Review

III. Matching (20pts). This section contains a collection of terms discussed in class in an “Answer Bank” (Table 1). Each question in this section describes a situation associated with an answer in the Answer Bank. Write the letter of the term in the Answer Bank that best describes each situation. Each answer in the Answer Bank will be used at most once.

16. (2pt) **E** Ada is assigned a user story. She creates a new branch, implements the necessary code, and then creates a pull request against the project’s main branch.
17. (2pt) **S** When Alan wants to integrate changes from his co-worker, he first needs to commit his own changes.
18. (2pt) **C** Grace is running a test-generation tool. It reports that there is a bug in her code, because it was able to cause the code to crash.
19. (2pt) **P** Donald is asked about a time that he had a disagreement with a co-worker.
20. (2pt) **L** Leslie spends 15 minutes each morning with his immediate team so that they can update each other on their progress.
21. (2pt) **G** Before Barbara commits a new method, a tool warns her about a non-idiomatic code pattern in her implementation.
22. (2pt) **I** Tim is assigned a user story. His first step is to write a test that currently fails.
23. (2pt) **B** Alonzo’s manager sends him a document that explains what he should implement, but not how to implement it.
24. (2pt) **F** Haskell’s job mostly consists of talking: to engineers, to customers, to upper management, and to other stakeholders.
25. (2pt) **T** Radhia is writing some code. As she does, Patrick sits with her and offers comments and/or critiques.

IV. Short answer (31pts). Answer the questions in this section in at most two sentences.

26. (3pt) Support or refute the following claim: estimating the time to build a software system is inherently harder than estimating the time to build a “physical” system, such as a highway. **Likely support. The cost of copying software is close to zero (unless the cost of copying physical infrastructure), so almost all new code will be innovative in some way. This increases planning costs. “It’s not research if you know it’s going to work.”**

27. Select 2 programming languages from the following list (circle your selections):

Java / C / TypeScript / Python

For each of the following axes, write 1-2 sentences comparing your two chosen languages along that axis. Each of your comparisons should reference at least one language feature of each language you are comparing.

- (a) (2pt) Performance: **(My answer concerns Java and Python; answers for other languages differ.) Python is interpreted and therefore has worse performance. Java’s JIT (HotSpot) makes it fast, especially for long-running programs like webservers.**
- (b) (2pt) Safety: **Java’s static type system prevents some errors at compile-time that would be run-time crashes in Python. Java also has a better verification ecosystem.**
- (c) (2pt) Developer Effort: **Python’s dynamic type system makes it easier to write prototype code quickly, since you don’t have worry about complex type conversions. Java also requires a lot of boiler-plate code to get started (consider `public static void main(String ↪ [] args)` vs `def main:...`)**

28. 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 both why the better choice performs better and why the worse choice performs worse.

- (a) (3pt) Regression testing better than fuzzing **Regression testing is better for preventing bugs that you’ve encountered before. Fuzzing finds bugs that you’ve never encountered.**
- (b) (3pt) Pair programming better than static analysis **Pair programming is better for making design decisions—whether at the code level (like variable names) or the architectural level, because humans are good at that and machines are not. Static analysis is better for remembering tricky edge cases.**
- (c) (3pt) Garbage collection better than static typing **Garbage collection is better when the program doesn’t need to be extraordinarily high performance: it trades some machine time while the program runs (to run the garbage collection) for some developer time (the developer doesn’t need to manually manage memory). If we don’t need performance for a particular program, encoding memory management in the type system (e.g., by writing in Rust) is a waste.**

29. Paul is a software engineer at Micro-softserve, a technology company that writes software for soft-serve ice cream machines. Paul's boss, Bill, reports that a large client of Micro-softserve (a fast food chain with a clown as a mascot) has been complaining that their ice cream machines are frequently not working, and Bill would like to argue to *his* boss that the software division is not at fault: Bill's team tests their code thoroughly! Paul is tasked with providing evidence that Bill can use to support this claim: that is, evidence that the software team's code is high-quality.
- (a) (2pt) Identify two specific quantitative metrics that Paul could report to Bill. **Two such metrics are statement coverage and mutation score; others are possible. A common mistake was saying "statement coverage and branch coverage". This got full credit on this part, but usually no points in the next two parts, because they're too similar and share almost all advantages and disadvantages as metrics.**
 - (b) (2pt) Write one sentence explaining an advantage of the first metric over the second. **An advantage of statement coverage is that it is easy to compute, and if the code is actually well-tested, it should be high.**
 - (c) (2pt) Write one sentence explaining an advantage of the second metric over the first. **An advantage of mutation score is that it is a better indicator of whether the tests will actually catch real bugs, because achieving high statement coverage doesn't actually require strong oracles.**
30. Arjun is a software engineer at LockMart, a company that provides high-assurance locks to the American government. Arjun is starting a new project in which he will write the software for a new electronic lock that will be used to improve security at nuclear reactors and other government locations where access control is highly-restricted.
- (a) (2pt) Name a quality requirement that is relatively *important* for Arjun's project, and then provide a one-sentence justification for why that quality requirement is important. **Correctness or security are the obvious choices: nuclear reactors and similar high-security facilities need guarantees that unauthorized personnel cannot enter.**
 - (b) (2pt) Name a different quality requirement that is relatively *unimportant* for Arjun's project—that is, a requirement on which compromise might be acceptable. Give a one-sentence justification for why it is okay to compromise on this quality requirement. **Usability is a good answer: government employees at such secure facilities require extensive training, anyway. Cost is also another good answer: high assurance is expected to be expensive anyway. Most quality requirements are reasonable answers, as long as the justification is sensible.**
 - (c) (3pt) Identify a specific technical decision or specific technique (the decision or technique should have been discussed in class) that Arjun can make or use in his new project that will help him trade-off between the two quality requirements you chose in parts a) and b). Justify why this specific technical decision or technique trades off these quality requirements appropriately. **The**

easiest technical decision is choice of language. An appropriate language is one with strong safety guarantees such as Rust (or a verification-first language like Dafny, even) if Arjun is trading off cost or usability for correctness. It's unusual that Arjun has the opportunity to choose a language, so this is a good thing to reach for here, but other answers are possible: for example, "use lots of powerful static analyses" or "achieve 100% MCDC coverage" are also acceptable answers.

V. Document-based Questions (24pts). All questions in this section refer to a documents **A-C**. 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** (3 pages).

31. (1pt) What kind of document is document **A**? **“Code review” or “pull request” (“change list” and other synonyms)**
32. (6pt) Name two software engineering practices that the engineers in this document have done well. For each, cite a specific piece of evidence from the document to support your claim. **One point for each principle; 2 for the justification. There are many good choices. Here are a few:**
- **feature branch development: the first screenshot shows that this PR is coming from a fork’s feature branch**
 - **continuous integration/automated testing: the “GitHub Actions” post shows that they’re taking this seriously**
 - **code review: the comments are helpful and critical but polite. A common mistake was to say this twice in different words.**
33. (3pt) Which comment from an engineer is the least important? Give a one-sentence justification for your answer. **The one from liuml07 beginning with “nit”. “Nit” indicates that the comment is unimportant, and there is only one such comment.**
34. (3pt) Support or refute the following claim: the code in this document deserves extra scrutiny from a human beyond what would normally be required for a code change. **Likely support. The proposed change involves concurrency, which usually requires extra care from reviewers.**

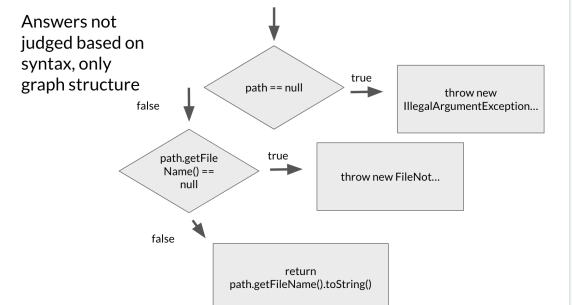
Questions on this page refer to Documents **B** and **C** (1 page). Document **B** is a warning from a static analysis tool. Document **C** is the code relevant to the warning.

35. (4pt) Support or refute the following claim: the static analysis warning in Document **B** is a false positive. **Likely support. As a human, we expect that `Path.getFileName()` is deterministic: that is, it returns the same value each time it is called. So, the null check on line 105 should be sufficient.**
36. (4pt) Rewrite the code in Document **B** so that a typical dataflow-based nullability analysis that checks for potential nullability problems will not issue a warning. (No partial credit: we will run FindBugs to determine if your answer elicits the false positive if we are unsure.) **Abstract the call to `path.getFileName()` into a local variable, so that its value cannot change after the null check.** E.g.,

```

1  protected String getFileName(Path path) {
2      if (path == null) {
3          throw new IllegalArgumentException("null path");
4      }
5      String fileName = path.getFileName();
6      if (fileName == null) {
7          throw new FileNotFoundException(path.toString());
8      } else {
9          return fileName.toString();
10     }
11 }

```



37. (3pt) Draw the control-flow graph for the code in Document **C**.

VI. “Your Choice” Reading Quiz (2 points).

Each question in this section is a reading quiz question for one of the “Your Choice” readings. Select **one** of the questions and fill in the question below (question number 38) with the letter of the question you are answering (and its answer, of course!). **DO NOT CIRCLE ANSWERS TO SUB-QUESTIONS IN THIS SECTION.** You may answer additional questions in the extra credit section (Section VII), if you have done more than one “Your Choice” reading.

38. (2pt) Write the letter of the question you are choosing to answer: **Any number.**

Write the answer to that question: **The answer.**

- (a) Ajami et al.’s *Syntax, predicates, idioms - what really affects code complexity?* Which of the following is one of the main findings in the article:
- A** **while** loops are more complex than **for** loops
 - B** **if** statements are more complex than **for** loops
 - C** **for** loops are more complex than **if** statements
- (b) Saff and Ernst’s *An Experimental Evaluation of Continuous Testing During Development*: The experiment described in the study was carried out using a modification or plugin to which editor:
- A** Emacs
 - B** Eclipse
 - C** Vim
- (c) Memon et al.’s *Taming Google-Scale Continuous Testing*: Of the 5.5 million tests that the authors considered, about how many had failed at least once?
- A** 63,000
 - B** 630,000
 - C** 6,300,000
- (d) Barr et al.’s *The Oracle Problem in Software Testing: A Survey*: Which of the following are categories of test oracle research identified by the paper? Circle all that apply.
- A** derived oracles
 - B** explicit oracles
 - C** ambiguous oracles
 - D** specified oracles
- (e) De Rosso et al.’s *Purposes, concepts, misfits, and a redesign of git*: **TRUE** or **FALSE**: as part of their evaluation, the authors performed a manual analysis of Reddit posts related to git.
- (f) Ernst et al.’s *The Daikon system for dynamic detection of likely invariants*: the second section of the paper contains a long example that implements a data structure in Java. Which data structure is it?
- A** queue
 - B** stack
 - C** list

- (g) Lamport's *Introduction to TLA*: Finish this quote from the reading: "A TLA formula is true or false on a **behavior**."
- A system
 - B** behavior
 - C logic
- (h) Anda et al.'s *Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System*: of the four companies considered in the study, three of the four were from the same country. Which country was it?
- A** Norway
 - B United States
 - C India
- (i) Behroozi et al.'s *Hiring is Broken: What Do Developers Say About Technical Interviews?*: the study considered comments from a website. Which website was it?
- A Reddit
 - B Stack Overflow
 - C** Hacker News
- (j) Bacchelli and Bird's *Expectations, Outcomes, and Challenges Of Modern Code Review*: **TRUE** or **FALSE**: based on the results of the study, reviews commonly detect deep, subtle, or "macro" level issues.
- (k) Hoare's *Hints on Programming Language Design*: The author believe that which of the following criteria are appropriate to consider in the choice of programming language? (Select all that apply.)
- A portability and machine independence
 - B** the language's support for designing, documenting, and debugging programs
 - C sponsorship by a rich and powerful organization
 - D a large and useful standard library and user community
- (l) Mokhov et al.'s *Build Systems a la Carte*: **TRUE** or **FALSE**: the authors claim that there are "two key design choices that are typically deeply wired into any build system" and that those design decisions are deeply coupled: the build system's choice in one seriously restricts its choices in the other.
- (m) Bessey et al.'s *A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World*: **TRUE** or **FALSE**: the authors claim that a not-understood bug report is commonly labeled a false positive, rather than spurring the programmer to delve deeper.
- (n) Ernst's *Notes on Program Analysis*: Which of the following is a component of the author's definition of an abstract interpretation (select all that apply):
- A program dependence graph
 - B** lattice
 - C typestate automaton

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

39. (1pt) In section III (Matching), there is a theme to the names used in the situation descriptions. What is the theme? **First names of famous computer scientists: Ada Lovelace, Alan Turing, Grace Hopper, Donald Knuth, Leslie Lamport, Barbara Liskov, Tim Berners-Lee, Alonzo Church, Haskell Curry, Radhia Cousot, and Patrick Cousot.**
40. (1pt) State something that you learned in this class so far that you didn't know before and that was a surprise to you. **Any reasonable answer that isn't covered in a lower-level NJIT CS class gets the point.**
41. (1pt) State something that we discussed in this class that you already knew or understood from a previous class or internship. Also, state where you learned this thing. **Any reasonable answer gets the point.**

For the remaining extra credit points, answer additional questions from Section VI ("Your Choice" Reading Quiz). You may answer up to five additional times. However, if you get *any* of these questions wrong (including the question in Section VI), you will receive *no credit for any "Your Choice" Reading Quiz questions*. So, you should only answer questions about readings that you actually did read!

42. (1pt) Write the letter of the question you are choosing to answer: Any letter.
Write the answer to that question: The answer.
43. (1pt) Write the letter of the question you are choosing to answer: Any letter.
Write the answer to that question: The answer.
44. (1pt) Write the letter of the question you are choosing to answer: Any letter.
Write the answer to that question: The answer.
45. (1pt) Write the letter of the question you are choosing to answer: Any letter.
Write the answer to that question: The answer.
46. (1pt) Write the letter of the question you are choosing to answer: Any letter.
Write the answer to that question: The answer.

Document A:

Fix unexpected sink side unsubscribe behavior #562

Merged Andy26 merged 3 commits into `master` from `andyz/fixSinkUnsubError` 3 days ago

Conversation 16 Commits 3 Checks 5 Files changed 7

Andy26 commented last week Collaborator

Context

There is a race condition on the subscription handler where the current state can be empty if the subscription is established before the handler service is started, which causes non-perpetual jobs to silently fail at startup.

- Added handling + logs to this case.
- Added extra test in integration test.
- Removed sinkSubscriptionHandlerFactory on TaskExecutor level (not used anymore).

Checklist

- `./gradlew build` compiles code correctly
- Added new tests where applicable
- `./gradlew test` passes all tests
- Extended README or added javadocs where applicable

github-actions bot commented last week · edited

Test Results

545 tests	+1	537 ✓ +1	6m 46s ⌚ - 1m 8s
128 suites	±0	8 ±0	
128 files	±0	0 ±0	

Results for commit [7210270](#) . ± Comparison against base commit [49cd280](#) .

▶ This pull request **removes** 1 and **adds** 2 tests. *Note that renamed tests count towards both.*

This comment has been updated with latest results.

liuml07 approved these changes last week [View reviewed changes](#)

liuml07 left a comment Contributor

LGTM but a second review is appreciated. I have limited context.

```

...is-runtime-loader/src/main/java/io/mantisrx/runtime/loader/SubscriptionStateHandlerImpl.java
va
... @@ -53,7 +53,9 @@ class SubscriptionStateHandlerImpl extends AbstractScheduledService impleme
53 53
54 54     @Override
55 55     public void startup() {
56 -         currentState.set(SubscriptionState.of(clock));
56 +         if (currentState.get() == null) {
    
```

liuml07 last week Contributor ...

is this if check needed?




```

...is-runtime-loader/src/main/java/io/mantisrx/runtime/loader/SubscriptionStateHandlerImpl.java
... @@ -81,11 +83,20 @@ protected void runOneIteration() {
81 83
82 84     @Override
83 85     public void onSinkUnsubscribed() {
86 +         if (currentState.get() == null) {
    
```


liuml07 last week Contributor ...

These two operations in the method i.e. get-then-update are not atomic, is it ok?



Andy26 4 days ago Collaborator Author ...

currentState should never be unset back to null in this instance, thus no race on getting non-null then reading null.



```

mantis-runtime/src/main/java/io/mantisrx/runtime/executor/SinkPublisher.java
109 -         if (onUnsubscribeAction != null)
110 -             onUnsubscribeAction.call();
111 -     }
101 +         .doOnCompleted(() -> logger.info("Sink observable subscription completed."))
    
```

liuml07 last week Contributor ...

nit: is other information useful in the log like subscribe etc.




Andy26 4 days ago Collaborator Author ...


subscribe?




```
...is-runtime-loader/src/main/java/io/mantisrx/runtime/loader/SubscriptionStateHandlerImpl.java Outdated
va
...   ...   @@ -53,7 +53,10 @@ class SubscriptionStateHandlerImpl extends AbstractScheduledService impleme
53   53
54   54   @Override
55   55   public void startup() {
56   -   currentState.set(SubscriptionState.of(clock));
56   +   if (currentState.get() == null) {
```


 **sundargates** 3 days ago Contributor ...

I would convert this into a Precondition instead.



 **Andy26** 3 days ago Collaborator Author ...

I don't think it has to fail if the service got started again. Let me remove the check here directly (atomic set with null should be sufficient.)



Document B:

on Foo.java:108:

```
[warning] findbugs:NP_NULL_ON_SOME_PATH_FROM_RETURN_VALUE
```

Style - Possible null pointer dereference due to return value of called method

The return value from a method is dereferenced without a null check, and the return value of that method is one that should generally be checked for null. This may lead to a NullPointerException when the code is executed.

Document C:

Foo.java:

```
101     protected String getFileName(Path path) {
102         if (path == null) {
103             throw new IllegalArgumentException("null path");
104         }
105         if (path.getFileName() == null) {
106             throw new FileNotFoundException(path.toString());
107         } else {
108             return path.getFileName().toString();
109         }
110     }
```