minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

General Reminders

- You **may** use *free* generative AI tools on this exam.
- The exam is open note, open lecture recordings, open course webpage, open computer (including Visual Studio, Python, etc.), and open Internet in general. You may use general webpages (e.g., Stack Overflow, Wikipedia, etc.).
- You may **not** use live help from another human.
- If you leave a sub-question entirely **blank** then you will receive one third of its points rounded down. For example, if a sub-question is marked "(4 points)", you would receive 1 point if you left it entirely blank.
- If you are asked to support or refute a position, indicate which option you selected. One word can suffice.
- If you are asked to provide a quote from multiple possible sources (e.g., either the lecture slides or a reading), indicate which source you selected.
- If you quote from a lecture slide, you must indicate which slide number. If you quote from something spoken verbally (out loud) during a lecture, you must include a timestamp.
- If you are asked to provide a quote to help support or justify your argument, your quote must be directly relevant to your argument.
- If we cannot verify that your quote is from one of the required sources (or if it appears to be fabricated), you will receive **no points** for that entire question.

Question 1. Word Bank Matching (1 point each, 15 points total)

For each statement below, input the letter of the term that is *best* described or the concept that is *most* related. Note that you can click each word (cell) to mark it off. Each word is used at most once.

A. — Causation	B. — Complexity	C. — Composite Design Pattern	D. — Correlation
E. — Creational Design Pattern	F. — Delta Debugging	G. — Designation Conflict	H. — Elicitation
I. — Fault Localization	J. — Informal Goal	K. — Large Language Model	L. — Maintainability
M. — Productivity	N. — Profiling	O. — Program Synthesis	P. — Risk
Q. — Stakeholders	R. — Static Analysis	S. — Validation	T. — Weak Conflict

Q1.1: **D**

A descriptive model of software readability may include this type of relationship between blank lines and readability.

Q1.2: **O**

Iroh asks a tool, such as ChatGPT or GitHub Copilot, to generate source code. The code may be reviewed by humans, but it was created by an algorithm.

Q1.3: **N**

Ozai is directed to reduce memory consumption in a program. Ozai pursues this by instrumenting each method to determine how much memory it allocates at run time, and second by analyzing the results after running the program.

Q1.4: **J**

Andrew is writing software for a microphone company. The company's market research suggests that customers want the audio to "not sound too breathy". This information, without elaboration, is provided to Andrew to help guide software development.

Q1.5: **P**

This planning factor includes both the odds of an event happening and also the consequences of that event happening.

Q1.6: **L**

GlazeBook programmers conclude that the new software they develop is likely to exist for some time. They structure their software to simplify subsequent changes, including both fixing bugs and also adding new features.

Q1.7: **Q**

The University of Michigan is considering revamping Wolverine Access. Lawyers, representatives from human resources, professors, and students are all gathered to provide input.

Q1.8: **E**

Aang structures a class so that the normal constructor cannot be called and another method must be called instead. This allows type information to be hidden.

Q1.9: **A**

This sort of relationship between software factors and outcomes provides the most confidence in predictions about the future. It is desired in normative models.

Q1.10: **S**

Gojo uses a series of interviews, walkthroughs and checklists to ensure that the requirements are complete and consistent.

Q1.11: **K**

A tool, such as a neural network transformer, that can create text in response to natural language prompts.

Q1.12: **G**

FloorMart's informal documentation uses the term "priority" for both operating system scheduler precedence and also for a queue data structure used in shortest path calculations. This confuses developers.

Q1.13: **B**

This attribute of software is commonly computed statically, but popular metrics associated with it may not align with human perceptions or judgments.

Q1.14: **C**

Aamir is writing a program that has access to video files and wants to use an analysis library that operates on sets of images. By structuring the code to treat individual frames and group of frames uniformly, maintainability is improved. From the code's perspective, the analysis library can now be applied to sets of frames.

Question 2. Guest Lectures and Concepts (16 points)

Reminder: While not all guest lecturers were able to make their PDF slides available, all of the lecture recordings are available at https://leccap.engin.umich.edu/leccap/site/1gz3vacgfuusnfw31vh and your notes summarizing and organizing the content of those guest lectures are available to you. If you include a spoken quote from a lecture, you **must include a timestamp** so that the graders can verify it.

(a) (4 points)

Consider the guest lecture by Zak Fry with a particular focus on Slide 16 of https://eecs481.org/lectures/ZFGuestLecture_public.pdf discussing exploiting cognitive biases.

Dr. Fry was targeting a particular type of defect. Using the terminology from class, would you expect these to be low, medium or high *severity* defects and why? (Use at most two sentences.)

Summarize *cognitive load* in one sentence, and then include a quote from one of our course readings or lecture slides to support or elaborate on it. Are Dr. Fry’s results most relevant to Formal Code Inspection, Passaround Code Review, both, or neither? (Beyond the quote, use at most three sentences.)

Your answer here.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

ANSWER:
Dr. Fry was targeting cybersecurity vulnerabilities (see “attacker” on slide 12, “suspicious traffic” objective on slide 14, etc.). Security bugs are classic examples of high severity issues. He detailed this verbally. In addition, we have indirect evidence (e.g., the government was paying a large amount of money for the work, which is rare for low-severity, low-priority topics).

We covered Cognitive Load in places like https://eecs481.org/lectures/se-18-brain.pdf (e.g., "The brain works harder (cognitive load)") and in the productivity lecture https://eecs481.org/lectures/se-19-productive.pdf . Dr. Fry’s task is more related to Code Inspection. In his task, participants are shown an already-existing system and are asked to look through it -- potentially through all of it. There are no new code changes or pull requests that come in for participants to judge. This may seem like a strange lens for Code Inspection: intuitively, there’s no way someone would have time to read the whole system. Indeed, that is a weakness we identified for Code Inspection (Slide 43, it’s very labor intensive, etc.) and Dr. Fry’s slides even address this: “TASK: Identify a critical file on the target machine (impossible)”. However, other reasoning to support Code Inspection (such as by linking its properties to cognitive load's properties) is possible. For example, some students called out that Formal Code Inspection is usually done under time pressure (Slides 60 and 61), which aligns with Fry's experiments.

It is true that Dr. Fry's research wouldn't hurt Passaround Code Review: that activity does also involve humans, cognitive biases, and the like. However, the question asked about the results presented in the Guest Lecture, and those were specifically about a Code Inspection-esque security-focused activity in which participants looked at an already-complete system, not at code checkins.

Some students expressed confusion at the "Stars" column in Dr. Fry's slides. Those represented how easy it was to research or operationalize the bias, not the severity of the associated defects that stem from those biases. The security defects are all high severity even if a cognitive bias is hard to control in the lab (low Star count).

1. +1 Correctly identifies these as high-severity defects with a brief valid justification.
2. +0.5 Identifies the defects as medium severity instead of high severity.
3. +1 Provides one accurate sentence defining cognitive load.
4. +1 Includes a relevant quote from course readings or lecture slides.
5. +1 Correctly identifies Formal Code Inspection with brief reasoning.
6. +0.5 Incorrectly identifies "both" Inspection and Review (instead of just Inspection), with brief reasoning.
7. 0 overall if the quote is fabricated or not taken from an approved course source.

(b) (4 points)

Consider the guest lecture from Arthur Krieger and Maya Vijan. In particular, around 44 minutes of the recording, Maya Vijan mentioned “Being able to communicate what is going on in a technical problem to a non-technical audience”.

On slide 7 of “Requirements, Validation and Risk” https://eecs481.org/lectures/se-14-valid.pdf we gave examples of stakeholders for the NASA Near Shoemaker mission. Identify one stakeholder in that diagram that would be categorized as “non-technical” in this context and explain why communication with that stakeholder would be critical. (Do not pick the asteroid. :-)) Use at most two sentences.

Choose a conflict type identified on slide 28 of “Requirements, Validation and Risk”. Name the conflict type, make up an example of that conflict type from the particular domain Krieger and Vijan are working in (be specific; general answers will not be accepted), and indicate how a software engineer and a non-technical audience might end up with inconsistent interpretations. Use at most two sentences.

Your answer here.

ANSWER:
A critical non-technical stakeholder is “Congress” (lower left), which provides the budget for NASA and thus for the mission. If goals and constraints are not clearly communicated to that non-scientific audience, the scientific goals will not be carried out because funding will not be available.

For conflicts, multiple examples are possible, noting that Charles Schwab is in the particular domain of banking and finance. So general answers about time or memory or the like will not be accepted. Example: *Structure clash*. “latest filing date” for taxes or financial disclosures as time point (e.g. Fri 5pm) vs. time interval (e.g. Friday). A technical audience might assume a particular time point, while a non-technical customer might assume that any time during that day is fine.

Another example might be a *designation clash* on a term like "account". To the software developers at the company, that may refer to a user account (e.g., with associated login credentials). To the finance people at the company, that may refer to a

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

repository of money (e.g., checking account, savings account, etc.).

1. +1 Identifies a correct non-technical stakeholder from the slide and briefly explains why communication with them is critical.
2. +1 Names a conflict type from slide 28 and provides a domain-specific example tied to Krieger and Vijan's banking/finance context.
3. +1 Explains how a software engineer and a non-technical audience might form inconsistent interpretations of that example.
4. +1 or +.5 Response respects constraints (specific to domain, not generic).
5. 0 overall if the stakeholder is invalid, the conflict type is not from slide 28, or the "specific domain" requirement is ignored.

(c) (4 points)

Consider Alex Collier's guest lecture and the story he told around slide 10 (https://eecs481.org/lectures/collier-guest.pdf) of a performance issue related to REDcap (a secure tool for storing medical data in a legally compliant way). He indicates that his final solution involved "Python multithreading for concurrent processing". Support or refute the claim that a dynamic analysis like Delta Debugging (see slides 10 or 61 of https://eecs481.org/lectures/se-12-debug.pdf for how Delta Debugging might apply to thread schedules, etc.) or sampling-based profiling (see near slide 48 of https://eecs481.org/lectures/se-11-faultloc.pdf) would have helped get him in that trouble spot. Include a quote from the slides or readings (your choice, but indicate which one) to support your argument. (After copying the quote, use at most four sentences.)

Your answer here.

ANSWER:
Almost certainly refute. While Delta Debugging can be applied to performance bugs (e.g., by defining Interesting() to be "takes longer than X seconds" or the like), it is a much more natural fit for functional correctness properties (e.g., crashing).

Sampling-based profiling is also unlikely to help in his situation. Sampling-based profiling requires running your program on its workload at least once -- and usually with a slowdown. By the time Alex had received the note from management saying "please stop it immediately", it was too late to run things again! DD also suffers from this problem, since it runs your entire program/workload multiple times. This really brings a weakness of dynamic analyses (compared to static analyses) to life: as a new hire, Alex is not in a position to monopolize a shared server re-running a slow program. Dynamic analyses require that you run the program.

Others might point out that sampling-based profiling can miss periodic behavior (quote from slides: "Can completely miss periodic behavior (e.g., you sample every k seconds but do a network send at times 0.5 + nk seconds)") and that the API calls that Alex was dealing with could well be regular or periodic. So it is unlikely to help Alex.

1. +1 States a refute position: Delta Debugging or sampling-based profiling would not have helped in Alex's situation, with a brief argument.
2. +0.5 Incorrectly states support (instead of refute), with a brief argument
3. +1 Correctly describes at least one limitation of Delta Debugging for this case (e.g., better suited to functional bugs, requires many re-runs, schedule sensitivity).
4. +1 Correctly describes at least one limitation of sampling-based profiling for this case (e.g., requires running the real workload again, misses periodic behavior, added slowdown, "too late to rerun").
5. +0.5 After incorrectly stating support (instead of refute), correctly describes one real benefit of DD or Profiling for this case.
6. +1 Includes a relevant quote from the specified slides/readings and identifies the source (which slide deck or reading it came from).
7. 0 overall if the quote is fabricated or not taken from an approved course source.

(d) (4 points)

Consider Henry Beckstein's guest lecture about working at Subaru Labs in Japan. At around the 62 minute mark in the recording (displayed as 1:02 in the UI), a student asks him a question about how he retrains himself or gains mastery in relevant concepts. Henry describes a technique ("Instead of looking at the problem …") that he has found effective.

Henry's answer closely aligns with one of the first five readings from our Productivity lecture (listed in the outline on slide 4 of https://eecs481.org/lectures/se-19-productive.pdf). Identify the reading and indicate how Henry's approach matches what we learned. Use a quote from Henry (exact spelling is not required, but include a rough timestamp for the graders; use the "CC"

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

button to turn on captions if you are not certain) and also a quote from the Productivity slides or readings to support your answer. (After copying the two quotes, use at most three sentences.)

Your answer here.

ANSWER:
Henry's discussion aligns very strongly with Chi et al.'s *Expertise in Problem Solving*. The slides note that "novices are more dependent on surface features, whereas experts focus more on the underlying principles".

Henry quotes: "I tried to group the problems by what was similar." "And so you think … maybe this problem is about fitting geometry … It may actually be about pigeonhole." "After you solve these problems enough … it's this solution because of this commonality."

That is, Henry is trying to avoid surface features (like "fitting geometry" or "square peg in round hole") and instead looking for underlying principles in common ("actually be about pigeonhole"). This is analogous to experts in physics trying to avoid surface features (like "these problems both involve pulleys") and instead looking for underlying principles in common ("conservation of energy").

1. +1 Correctly identifies which of the first five Productivity readings Henry's answer aligns with ("Chi et al." or "Expertise in Problem Solving").
2. +1 Explains how Henry's described technique matches the key idea from that reading.
3. +1 Includes a quote from Henry with an approximate timestamp.
4. +1 Includes a relevant quote from the Productivity slides or readings.
5. 0 overall if either quote is fabricated or not taken from an approved course source.

Question 3. Delta Debugging (13 points)

Answer the following questions about the Delta Debugging algorithm.

(a) (5 points)

We discussed three key assumptions for the Delta Debugging algorithm: Monotonic, Unambiguous, and Consistent (https://eecs481.org/lectures/se-12-debug.pdf). Your company is tasked with finding a small Interesting subset of a customer dataset using Delta Debugging. For this customer's dataset and use case, Delta Debugging is Consistent. However, instead of mentioning Monotonic and Unambiguous, the customer reports that their dataset and notion of Interesting satisfy a **Growth** property:

- Forall X, Forall Y, Forall Z. Interesting(X) && Interesting(Z) → Interesting(Y ∪ (X ∩ Z))

Support or refute the claim that Delta Debugging will work correctly with the Consistent and Growth assumptions. Include a quote from the slides or readings (your choice, but indicate which one) to support your argument. (Beyond copying the quote, use at most four sentences.)

If your browser does not display the mathematical symbols correctly, here is another equivalent rendering of Growth: `Forall X, Forall Y, Forall Z. (Interesting(X) [AND] Interesting(Z)) [IMPLIES] Interesting(Y [UNION] (X [INTERSECTION] Z))` .

Your answer here.

ANSWER:
Support. Growth logically implies Monotonic and Unambiguous. So Consistent and Growth together logically imply all three of Delta Debugging's required assumptions.

Since Growth is universally quantified, you can instantiated X, Y and Z with any values you like. If you pick X=Z and Y={c}, you get Monotonic. If you pick Y={}, you get Unambiguous.

Quotes from Slide 55 (etc.) describing those hypotheses are appropriate.

1. +1.5 States a Support position on whether Delta Debugging will work under Growth + Consistent.
2. +1 Provides a correct argument about how Growth relates to the Monotonic assumption. This might include using correct logical reasoning with the Growth property (e.g., appropriate instantiations of X, Y, Z) to justify the claim.

3. +1 Provides a correct argument about how Growth relates to the Unambiguous assumption (as above).
4. +1 Includes a relevant quote from the readings or slides and identifies the source.
5. +0.5 Writing satisfies the prompt constraints (less than or equal to four sentences after the quote; reasoning is coherent and on-topic).
6. 0 overall if the quote is fabricated or not from an approved course source.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

(b) (3 points)

Consider the set suite of PNG files you used in Homework #5: https://eecs481.org/hw5/large-png-suite.zip. A customer wants you to find a small Interesting subset of that test suite, where `Interesting` is defined as “the total file size of all files in the subset is at least 8799 bytes”. The total file size of a set of files is the sum of the individual sizes. Reminder: You can use or review your Delta Debugging implementation from class.

What is a minimal Interesting subset of that test suite? Format your answer as a Python list of strings. Example: `[ "1.png", "2.png" ]`

Scripting hint. You can use shell commands like “`cat … | wc -c`” to quickly determine the total file size of multiple files and shell operations like “`if [ x -ge y ]`” to quickly determine if x >= y. (You are not required to do so; this is just a reminder.) Example:

```
$ cat 1.png 2.png 3.png | wc -c
2053

$ if [ `cat 1.png 2.png 3.png | wc -c` -ge 2000 ] ; then echo yes greater than 2000 ; else echo not big enough
; fi
yes greater than 2000
```

Your answer here.

ANSWER:
[“397.png”] (or one of 82 other single-file answers). Notably, a key concept here is that you don’t actually need DD at all to solve this: a greedy approach, where you just sort the files by size and include them until you exceed the threshold, works. There are actually 83 single files in that test suite that are 8799 bytes or longer alone. Any one of them is a correct answer here. Combinations of two or more files are not correct, since the problem asks you for a minimal Interesting subset, not a subset that DD returns. That is, the way this question is phrased, you have to be perfect even if DD is not.

Students could arrive at the right answer by running their `delta.py` code from HW5 (or any other source) until it produces a correct answer, or by reasoning about the problem and looking at the contents of the archive (e.g., `ls -la --sort=size`), realizing that many single files are above 8799, and picking any one of them.

This problem's structure was intentionally reminiscent of HW5c, in which students were also given a large minimization problem to solve with DD, only to determine, via investigation, that the setup lacked one of the critical DD assumptions.

(c) (5 points)

We continue the scenario from the previous subquestion. Support or refute the general claim that Delta Debugging will work correctly for that notion of Interesting. Include a quote from the slides or readings (your choice, but indicate which one) to support your argument. (Beyond copying the quote, use at most four sentences.)

Your answer here.

ANSWER:
Refute. This setup violates the Unambiguous assumption, so DD may not find an Interesting subset in general (slide 56 of https://eecs481.org/lectures/se-12-debug.pdf). Slide 59 also gives an example of DD returning an Interesting-but-not-minimal subset if Unambiguous is not true.

To see how this setup violates Unambiguous, consider three hypothetical files *for the purposes of contradiction*: "size9000", "size9001", and "size5" — with sizes 9000, 9001, and 5 respectively. The set { size9000, size5 } is Interesting (total size 9005 > 8799). The set { size9001, size5 } is Interesting (total size 9007 > 8799). If Unambiguous were true, then the intersection of those two Interesting sets would be Interesting. But their intersection is just { size5 }, which is not big enough alone to be Interesting (5 < 8799). This is a contradiction, so Unambiguous is false in this setting.

By contrast, this setting maintains the Monotonic assumption (if you're already 8799 bytes, adding another file does not decrease that: there are no negative file sizes). This problem also maintains the Consistent assumption (file sizes are not "Unknown"). So Monotonic and Consistent are not the reason that DD fails. Instead, the reason that DD will fail is that the setup is not Unambiguous.

1. +1.5 Clearly states a Refute position on whether Delta Debugging will work for this notion of Interesting.
2. +2 or +1 Correctly identifies which Delta Debugging assumption fails (e.g., Unambiguous) AND explains why.
3. +2 or +1 After incorrectly stating Support, correctly identifies how two assumptions are present in this setting (Consistent and Montonic)
4. +1 Includes a relevant quote from the slides or readings and identifies the source.
5. +0.5 Writing satisfies the prompt constraints (less than or equal to four sentences after the quote, coherent reasoning).
6. 0 overall if the quote is fabricated or not from an approved course source.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

Question 4. Design and Maintainability (10 points)

Answer the following questions about design and maintainability.

(a) (5 points)

Consider Liu et al.'s "Automatic Generation of Pull Request Descriptions" https://eecs481.org/readings/automaticpr.pdf. In class, we discussed **What** and **Why** as categories of documentation. Support or refute the claim that Liu et la.'s special handling of out-of-vocabulary (OOV) words will allow their approach to generate higher-quality Why information for pull request summaries. Give an example from the slides of the sort of Why documentation that you think would be handled well (or not) by Liu et al.'s method and indicate which slide number it corresponds to in https://eecs481.org/lectures/se-17-designmaint.pdf. Include a quote from the Liu et al. paper to support your argument. (Beyond copying the quote, use at most four sentences.)

Your answer here.

ANSWER:
Refute. This can be answered using only the slides and text up through Section II-B of the paper (the required part of that reading) and the slides. While their OOV approach does help with comment synthesis, it only helps with words that are present in the code. Paper quote: "We observe that the OOV words in a PR description can often be found in the corresponding "article"." Why-style comments like "management is worried about buffer overruns" from Slide 16 won't have corresponding mentions in the source code (e.g., management is not an identifier name). So OOV won't be able to infer or synthesize it.

Students might also reference details of their OOV handling, such as its "pointer generator" nature: "So we integrate the pointer generator [16] in our approach to solve this problem. With this component, our approach can either select a token from the fixed vocabulary or copy one from the source sequence at each decoding step" from Page 4. The details about "pointer generator" aren't as relevant for us, but that quote does show it won't work for Why documentation because of its "fixed vocabulary".

1. +1 Clearly states a Refute position on whether Liu et al.'s OOV handling improves the generation of Why information.
2. +1 Provides a correct argument about what OOV handling can or cannot capture (e.g., limited to words appearing in the code, not external rationale).
3. +1 Includes a relevant quote from the Liu et al. paper and identifies it as such.
4. +0.5 Gives an example of Why documentation from the specified lecture slides and correctly identifies the slide number.
5. +1 Correctly identifies why their example of Why documentation from the slides would be handled well or not
6. +0.5 Writing respects the prompt constraints (four sentences after the quote, reasoning is coherent and specific).
7. 0 overall if the quote is fabricated or not from an approved course source.

(b) (5 points)

Consider the driver, mutate and subject Python files from the Homework #3 exercise https://eecs481.org/hw3/hw3.zip. Consider the **Observer** (or publish-subscribe) software design pattern. Support or refute the claim that the driver should use the Observer design pattern to improve maintainability when running each generated mutant against the test suite. A subscriber could receive a push-style update() for each test outcome and note when exactly one test has failed for a mutant (and thus that the mutant is neither too strong nor too weak in that Homework's definition). Include a quote from the slides or course readings (your choice, but indicate which one) to support your argument. (Beyond copying the quote, use at most four sentences.)

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

Your answer here.

ANSWER:
Refute. This is a synchronous, procedural control flow with a one-to-one dependency. You only have one kind of observer. The logic is simple and centralized. Having observers is unlikely to reduce coupling or improve flexibility. The added complexity (overhead, observer registration, notification management) may outweigh any potential benefit.

Slides quote (Design for Maintainability, slide #38): "Introduces extra interfaces and classes: code can be harder to understand; adds overhead if the strategies are simple." Wiki Reading Quote: “Define a one-to-many dependency between objects where a state change in one object results in all its dependents being notified and updated automatically.” We only have 1 dependent, so ”all of its dependents” is overkill.

1. +1.5 Clearly states a refute position on whether the driver should use the Observer pattern.
2. +2 or +1 Provides a correct argument about why the Observer pattern is not appropriate (e.g., only one dependent, simple centralized logic, procedural flow, added complexity/overhead outweighs benefits).
3. +1.0 (Max) Alternatively, incorrectly argues for using the Observer pattern but uses claims that are true about the Observer pattern
4. +1 Includes a relevant quote from the slides or course readings and identifies the source.
5. +0.5 Writing respects the prompt constraints (four sentences after the quote; reasoning is coherent and specific).
6. 0 overall if the quote is fabricated or not from an approved course source.

Question 5. Requirements and Elicitation (10 points)

Answer the following questions about requirements and elicitation.

(a) (5 points)

Consider the notion of a **verifiable non-functional requirement** from slide 38 of https://eecs481.org/lectures/se-13-req.pdf. Support or refute the claim that the Homework #2 specification at https://eecs481.org/hw2.html contains a verifiable non-functional requirement that the student (you) is instructed to follow. If you choose to support, quote the text showing the requirement and justify your answer. If you choose to refute, quote the text showing a requirement that comes close to qualifying but officially does not, and justify your answer. (Beyond quoting from the text, use at most four sentences.)

Your answer here.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

ANSWER:
Refute. While the HW2 specification mentions times (e.g., “it took 6.3 hours to run the AFL tool”) and numbers (e.g., “our scatterplot must include data reaching at least 510 paths_total on the y axis”) those are not verifiable non-functional requirements. The former (6.3 hours) is not a requirement at all. The latter (510 paths) is a requirement, but it is a functional requirement. It does not specify the manner or quality in which the answer is delivered (e.g., performance, reliability, response-time, memory usage). Other aspects of the specification (e.g., you should “Elaborate and reflect” and provide a “convincing” analysis that “shows non-trivial insight”) are quality requirements but they are not verifiable as written. (Indeed, that ambiguity is often a source of student stress!)

Some students also called out quotes like "If you have a different number of classes, you are fine." or "Regardless of when you finish, everything is fine." which are, at best, rough estimates for students. Indeed, "Regardless of when you finish, everything is fine" is the opposite of a firm, checkable numerical timing constraint. These are not rules about the manner in which the answer must be delivered in measurable terms.

Partial credit would be given for stating Support and identifying something like "you must start it days before the due date to be able to complete it in time" as a non-functional (timing) requirement that the student must follow during development. However, that property is verifiable (e.g., we could ask for logs or screenshots showing that you started days before the due date, and not just in the last day: the time quantification makes it measurable and checkable), even though we did not verify it. The question asks whether the property is verifiable, not whether it was verified in practice.

While grading the exams, a few students actually caught a very minor instance that could be seen as a verifiable non-functional requirement. The Randoop instructions use the following example:

```
$ java -classpath ${RANDOOP_JAR}:target/classes/ randoop.main.Main gentests --classlist=classes.txt --time-
limit=120
```

A numerical measured time limit (here enforced by a timer in Randoop) would be a verifiable non-functional requirement that the student's work (i.e., programs run by the student) would follow. These installation details are broadly optional — you have to click to see them, and they are near directives saying you should try it yourself rather than just following these, etc. — but many students did interpret them as strict requirements. So while we believe the primary answer is Refute (this 120 second limit is optional, you could run for 180 and still get full credit on the report, etc.), students who mentioned Support while pointing to *this and only this exact time limit* also earned full credit.

Finally, some students expressed slight conceptual confusion by suggesting that the "510 paths_total" was a constraint on how long AFL had to run. However, while it may have felt like that subjectively, it's not actually a constraint on time (e.g., one person might run for an hour to get 510 paths, another might run for 10 hours to get 510 paths). The 510 paths_total is best understood as a functional property, rather than a constraint on the manner in which the answer is delivered (like power consumption or privacy or availability).

1. +1 States a clear Refute position.
2. +1 Alternatively, states a Support position calling out the 120-second Randoop limit
3. +0.5 Incorrectly states a Support position, identifying a verifiable (but not verified) non-functional requirement
4. +0.5 Incorrectly states a Support position, identifying a verifiable functional requirement (like 510 paths_total) instead of a non-functional one
5. +1 Includes a quote from the HW2 specification that is relevant to the argument (it may be imperfect or only loosely related — as long as it is a real quote, give credit).
6. +1 Gives a reasonable explanation for why the HW2 text does or does not qualify as a verifiable non-functional requirement.
7. +1 or + 0.5 Makes connection to the definition from slide 38
8. +0.5 Writing respects most of the prompt constraints .
9. 0 overall only if the quote is fabricated or not from an approved course source.

(b) (5 points)

Consider the concept of a **Conflict** from requirements elicitation https://eecs481.org/lectures/se-14-valid.pdf. Consider the Homework #1B specification at https://eecs481.org/hw1.html and the Bjarnason et al. reading at https://eecs481.org/readings/bjarnason-re-vv.pdf. Support or refute the claim that RE conflict resolution on the word “error” would likely result in higher coverage for student test suites in HW1B. Include a quote from the reading and a quote from the specification to support your argument. (Beyond copying the quotes, use at most four sentences.)

Your answer here.

ANSWER:
Support. This idea has both theoretical and practical (481-specific) support. For theory, the third mention of “coverage” in Bjarnason et al. is “rich interactions between RE and testing can lead to pay-offs in improved test coverage ”. RE conflict resolution for a testing assignment would be an interaction between RE and testing.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

For this particular activity, the HW1b spec uses “error” to refer to both “build process errors” (as in “(If you get an error like "bash: cd: libpng-1.6.34: No such file or directory",”) and also “PNG files with errors” (as in “It seems libppng handles png files which can have different "parameter values", errors etc. “). The former do allow the student to collect coverage information at all, but the latter are a key hint to obtaining higher coverage by exercising error-handling code by submitting “invalid” or erroneous PNG files. Testing for errors can improve coverage (e.g., slide 11 of https://eecs481.org/lectures/se-05-testquality.pdf , slide 18 of https://eecs481.org/lectures/se-04-qatesting.pdf , etc.) but this is not stated clearly in the HW1 specification (indeed, this is intentional: it’s a bit of a conceptual puzzle for 481 students to help reinforce concepts associated with coverage). Resolving that conflict -- such as by explicitly stating “build process error” vs. “erroneous-but-still-runnable PNG file” -- would directly point out a known-useful strategy for constructing high-coverage test suites to students. Concretely, it might lead to more coverage in files like `pngerror.c` which focus on error cases.

ChatGPT sometimes favors "Refute", claiming that without a concrete requirement, students have no obligation (or guidance) to design tests for malformed PNGs, which which touches on aspects relevant to the correct answer, but is not the right answer.

1. +1 States a clear and correct position (Support)
2. +1 Includes a relevant and appropriately chosen quote from the Bjarnason et al. reading (must clearly relate to coverage or RE/testing interactions; vague or unrelated quotes do not earn credit).
3. +1 Includes a relevant quote from the HW1B specification that directly shows the conflicting uses of the word “error” (generic or incorrect quotes do not earn credit).
4. +1.5 Clearly explains why resolving the “error” conflict would likely increase HW1B test-suite coverage (explanation must tie back to exercising error-handling behaviors or clarifying expectations).
5. +0.5 Writing satisfies all prompt constraints: after the quotes, no more than four sentences, and reasoning is specific and accurate.
6. 0 overall if either quote is fabricated or not taken from an approved course source.

Question 6. Fault Localization (11 points)

Answer the following questions about fault localization.

(a) (5 points)

Consider Table I of the Pearson et al. reference from Homework #5: https://eecs481.org/hw5/fault_localization_effectiveness_icse_2017.pdf. The two bottom-most rows report on over 3,000 real and artificial faults in programs like JFreeChart (seen in Homeworks #1 and #2). They note that “Our results (bottom) uphold previous results on artificial faults but not on real faults”.

What is an example of a previous result on artificial faults that they uphold? What formal software engineering concept (use its precise name) is generally called into question by the disparity between their results on artificial and real faults?

Include a quote from the relevant slides or course readings (your choice, but indicate which one) to support your argument. (Beyond copying the quote, use at most four sentences.)

Your answer here.

ANSWER:
An example of a result upheld: “Our results confirm 70% of previously-reported comparisons (such as “Ochiai is better than Tarantula” [26], [27], [31], [44], [49])” (from the Introduction on Page 1).

The formal software engineering concept that has its validity questioned by these results is the *Coupling Effect Hypothesis*. The CEH states that simple faults are coupled to complex ones such that test data that can detect simple faults will also detect complex faults. The CEH is commonly used in mutation analysis, suggesting that testing results from simple artificial faults (like first-order mutants) will generalize to testing on real-world complex faults. However, the Pearson et al. paper is reporting that some testing activities that detect simple artificial faults (in this case, Fault Localization) do not also work that well on real-world complex faults. Example quote: “Metallaxis is better than Ochiai on artificial faults [33], but Ochiai is better than Metallaxis on real faults”. (The results don’t strictly “disprove” the CEH, which is why the question uses indirect phrasing like ”called into question” and “generally”. But the real-world, general utility of something like the CEH would be that it strongly suggests that if Metallaxis performs well on simple artificial faults, it should also perform well on complex real-world faults, thus allowing you to make a software engineering process decision about whether or not to deploy it on your real-world software. But this paper is finding that sort of prediction to not hold true.)

Full credit will also be awarded for identifying the *construct validity* or *external validity* of fault localization or mutation analysis research. Just saying "validity" would not be enough, but the full two-word term "external validity" or "construct validity" would

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

be enough. We had originally intended only CEH to count, but were convinced by student answers to accept these two as well.

1. +1 Correctly identifies an example of a previous result on artificial faults that Pearson et al. say is upheld (e.g., ranking relationships such as “Ochiai is better than Tarantula”).
2. +1 Correctly names the formal software engineering concept called into question (must explicitly state “Coupling Effect Hypothesis” or "external validity" or "construct validity").
3. +0.5 Does not name one of those three, but instead calls out the general "validity" of FL or mutation
4. +1.5 Explains why the disparity between artificial and real faults challenges that concept (may reference the mismatch in performance or generalization; -0.5 per factually incorrect statements).
5. +1 Includes a relevant, accurate quote from the readings or slides and identifies the source. Quotes that are loosely paraphrased, irrelevant, or untraceable do not earn credit.
6. +0.5 Writing meets all prompt constraints: ≤ four sentences after the quote, clear reasoning, and no major factual errors.
7. 0 overall if the quote is fabricated or not from an approved course source.

(b) (6 points)

Consider the Barinel Fault Localization algorithm from the bottom right of page 3 of https://eecs481.org/hw5/fault_localization_effectiveness_icse_2017.pdf. Its formula is:

```
Suspiciousness(s) = 1 - (passed(s) / (passed(s) + failed(s)))
```

(Reminder: passed(s) is the number of passed test cases that executed statement s.) Consider the following Python code with certain statements identified by calls to print():

```python
def spider(x,y):
        print("1")
        if x < y:
                print("2")
        elif x > y:
                print("3")
        while x > 4:
                print("4")
                x = x / 2
                if (x == y):
                        print("5")
                        abort() # test fails if this is reached
        if x % 2 == y % 2:
                print("6")

```

With respect to the following six tests (i.e., test input values for x and y):

```
0,0
1,2
11,22
7,3
3,9
8,4
```

What is the Barinel Suspiciousness score for each identified statement? Format your answer as a Python list of numbers using at most 3 figures after any decimal point. Example: `[ 1.0, 0.667, -1.333, 0.0, 0.5, -1.0 ]`

If you submit a non-empty answer for this question, it must be entirely correct: **no partial credit** will be awarded. (If you do not submit an answer for this question, you receive the usual amount for leaving it blank.)

Your answer here.

ANSWER:
[0.167, 0, 0.5, 0.333, 1, 0] . This should admit automatic grading. One potential conceptual misunderstanding relates to test (11,22), which visits statement 4 twice (going around the loop). That does not “count twice” for fault localization. Since the only failing run is (8,4), only the statements visited on that run can have a non-zero suspiciousness. Generative AI may not reason about execution counts correctly.

```
** 0,0 visits 1 6
** 1,2 visits 1 2
** 11,22 visits 1 2 4 4
** 7,3 visits 1 3 4
** 3,9 visits 1 2 6
** 8,4 visits 1 3 4 5           # failing run

stmt 1 = 1 - (5/6) = .167
stmt 2 = 1 - (3/3) = 0
stmt 3 = 1 - (1/2) = 0.5
stmt 4 = 1 - (2/3) = .333
stmt 5 = 1 - (0/1) = 1
stmt 6 = 1 - (2/2) = 0
```

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

Question 7. Software Engineering Cognition (15 points)

Answer the following questions about cognition and software engineering.

(a) (6 points)

Consider the individual performance results from Sackman et al.'s *Exploratory Experimental Studies Comparing Online and Offline Programming Performance*, and from Dougherty and Thadani's *Economic Value of Rapid Response Time*. These are reproduced in the early slides of https://eecs481.org/lectures/se-19-productive.pdf. Consider Begel and Zimmermann's report from Microsoft, *Analyze This! 145 Questions for Data Scientists in Software Engineering* (https://eecs481.org/readings/datase.pdf).

Identify two areas related to individual productivity: one in which the Begel and Zimmermann report is in alignment with one of the two productivity readings and one in which it opposes them. For each, include a different quote from the slides or course readings (your choice, but indicate which one) to support your argument. (Beyond copying the quotes, use at most six sentences.)

Your answer here.

ANSWER:
Alignment. The Sackman reading reports (slide 17): "Correlations between all experimental measures, adjusted scores, grades, and the BPKT results were determined. … The results showed no consistent correlation between performance measures and the various grades and test scores." That is, those measures don't correlate with individual productivity (i.e., "it doesn't work"). The Begel and Zimmermann reading is in alignment with this, as shown on Table 3 on page 9: "Questions with the highest "unwise" percentage: Which individual measures correlate with employee productivity (e.g., employee age, tenure, engineering skills, education, promotion velocity, IQ)?" The participants also indicate that it would be "unwise" to use measures like education or IQ to correlate with individual employee productivity ("unwise" is diplomatic phrasing for "this doesn't work").

Another possible alignment is the emphasis on environment factors. The Rapid Response work notes "All users benefited from sub-second response time" (slide 11), while Begel and Zimmermann call out "impact their software development processes, practices, and tools" (Page 1, Introduction) — improving tools and processes could reduce response time.

Opposition. The Sackman reading reports (slide 18): "It is apparent from the spread of the data that very substantial savings can be effected by successfully detecting low performers. Techniques measuring individual programming skills should be vigorously pursued …" By contrast, Begel and Zimmermann find modern opposition to that recommendation: "We also observe opposition against the use of analytics to assess the performance of individual employees or compare them with one another."

You could also argue Opposition based on the Sackman reading framing productivity as an individual property, while the Begel and Zimmermann reading admits that it can be a group property or investigation (e.g., on Page 7: "Some respondents wanted to know how to monitor ... their team's productivity").

Incorrect Alignment. It may be tempting to report a strong alignment between Dougherty and Thadani's finding that "experienced engineer working with sub-second response was as productive as an expert with slower response" (i.e., that you can make novices as productive as experts by giving them faster machines) with references to "infrastructure" or "tools" in the Begel and Zimmermann paper. However, the Begel and Zimmermann paper is not talking about the same concepts. Their mention of "sharing tests and test infrastructures across teams" was not about reducing the response time of testing (cf. sharing tools and knowledge), and their mention of "What is the impact of tools on productivity?" is about software tools and not about machine response time. Generative AI may favor responses suggesting that tools, environments and infrastructure are all the same category, but the B&Z paper does not mention the word "environment" and is not talking about faster infrastructure.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

1. +1 Clearly identifies one area where Begel & Zimmermann align with either Sackman et al. or Dougherty & Thadani.
2. +1 Includes a relevant quote from the slides or one of the productivity readings supporting the alignment (quote must be real and from an approved source).
3. +1 Clearly identifies one area where Begel & Zimmermann oppose one of the productivity readings.
4. +1 Includes a different quote from the slides or readings supporting the opposition (must be real and properly sourced).
5. +1.5 Provides accurate and coherent explanations for both the alignment and opposition (explanations must meaningfully connect the concepts rather than merely restating the quotes ; -0.5 per factually wrong statements).
6. +0.5 Writing respects the prompt constraints (≤ six sentences after quotes, arguments are specific and factually correct).
7. 0 overall if any quote is fabricated or not from an approved course source.

(b) (5 points)

Consider McLoughlin et al.'s *Programmers' Visual Attention on Function Call Graphs During Code Summarization*, as described starting on slide 22 of https://eecs481.org/lectures/se-num-brain.pdf (and optionally available in full at https://yuhuang-lab.github.io/paper/huang-ase25.pdf). Then consider Graham et al.'s *gprof: a Call Graph Execution Profiler* https://eecs481.org/readings/gprof.pdf, which describes **flat** profiles in Section 5.1.

Support or refute the claim that flat profiles should be used to support developers during summarization. Include, as evidence, a quote from the slides or course readings (your choice, but indicate which one) to support your argument. (Beyond copying the quote, use at most four sentences.)

Your answer here.

ANSWER:
Likely refute. A critical concept is that a call graph profile is generally a "superset" of the information in a flat profile. A second key concept is correlation vs. causation (perhaps best viewed here as "absence of evidence is not evidence of absence").

In the gprof paper, "The flat profile consists of a list of all the routines that are called during execution of the program." More critically: "The major entries of the call graph profile are the entries from the flat profile, augmented by ..." In particular, the nodes in the call graph are the functions -- the same functions that would be listed in the flat profile.

The slides note "The more methods they looked at that were callers/callees → worse summary quality". A developer looking at a flat profile for guidance during summarization would see some methods that happen to be callers/callee (and McLoughlin et al. find that doing so lowers performance) and some methods that are not callers/callees (about which we have no information). Lacking any external reason to believe that non-callers/callees are specifically good , and instead just knowing that callers/callees are bad, a flat profile would behave more like a watered-down call graph profile. The user will still see some callers/callees, just fewer of them. (From a zoomed out view, the flat profile is like the call graph profile, but without the edges. Table V of the paper, which is not required reading and not necessary for a full answer, makes it clear that the nodes alone also correlate negatively with summary quality.)

Students could also just call out flat profiles as useless directly, arguing that they primarily highlight execution times, which are not the sort of information we've seen in class that people use (e.g., beacons, semantic cues, etc.) during comprehension activities (of which summarization is one example).

Because this question hinges on a potential cognitive bias, students who did not answer correctly are likely to have an intuitive reaction that flat profiles are (or "could be") good. "If caller/callee nodes are bad, and the flat profile wouldn't involve looking at them as often, doesn't that make the flat profile good?" Not necessarily: the question asks you about using flat profile at all (i.e., in isolation), not necessarily about using them compared to call graph profiles. "If caller/callee nodes are bad, perhaps the other nodes are good. It's possible that looking at them instead could help!" Yes, it is possible. But we don't have any evidence for that, and the question asks you about evidence. "If XYZ is bad, isn't doing something else, call it PQR, good?" It depends. XYZ is "robbery" and PQR is "donating to charity", then PQR is likely better. If XYZ is "robbery" and PQR is "murder", PQR is likely worse. The underlying concept here is that we know the call graphs are bad, we don't have any reason to believe the flat information is good, and we know the flat nodes are a subset of the bad call graphs (which isn't proof that they are bad, but certainly isn't evidence that they are good).

1. +1 Clearly states a Refute (or "likely refute") position on whether flat profiles should be used to support developers during summarization.
2. +1.5-+0.5 Explains the relationship between flat profiles and call-graphs in general (+0.5). Explains that flat profiles are essentially a subset or projection of the call-graph information from gprof (e.g., same nodes, no edges) or suggests that flat profiles focus on execution times (+1).
3. +1 Correctly connects the McLoughlin et al. finding that looking at caller/callee methods harms summary quality to why flat profiles are not good evidence-based support for summarization.
4. +1 Alternatively, correctly connects another course concept on code comprehension or summarization to explain why flat profiles are not good (such as from Siegmund et al.'s *Measuring Neural Efficiency of Program Comprehension*)
5. +1 Includes a relevant quote from the slides or course readings (e.g., gprof description, McLoughlin result) and identifies

the source.

6. +0.5 Writing respects the prompt constraints (≤ four sentences after the quote; reasoning is coherent and specific).
7. 0 overall if the quote is fabricated or not taken from an approved course source.

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

(c) (4 points)

You are working at a company that has collected COCOMO-style project time cost data. For this question, use the COCOMO numbers presented on Slide 34 of https://eecs481.org/lectures/se-02-risk.pdf.

You are managing a team that is being asked to make a "Very High" complexity project with a "Nominal" required turnaround time. In addition:

As part of your requirements elicitation process, you have conveyed prototypes to the customer that highlight tradeoffs in quality properties. The customer has reported that either of the following two configurations would be acceptable:

- A. The delivered software must comply with "Extra high" run-time performance constraints and with "Nominal" memory constraints
- B. The delivered software must comply with "High" run-time performance constraints and with "Extra high" memory constraints

Similarly, you have two options for how you will hire and train your team of developers (cf. slide 50 of https://eecs481.org/lectures/se-num-brain.pdf):

- C. You can hire developers with "Low" software engineer capability.
- D. You can hire developers with "Low" software engineer capability, but subject them to a language ability training course. This training course takes time and slows down project delivery by 1.25x, but has the overall effect of giving your team "High" software engineer capability.

Indicate which software process decisions you should make (A or B, C or D) to minimize the predicted time cost of the project. Format your answer as a Python list containing two strings and a number rounded to two digits after the decimal, where the rounded number represents the total COCOMO-predicted time. Example: `["A", "C", 9.12]` would mean "choose option A, choose option C, and the final time cost will be 9.12".

Your answer here.

ANSWER:
["A", "D", 2.32]

You should choose options A and D. This results in a project time cost of 2.3x the nominal previous projects at your company. This should admit automated grading.

```
1.66 <- want A instead of B
1.11 * 1.56 = 1.732

1.17
1.25 * 0.86 = 1.075 <- want D instead of C
```

… with the whole thing at "Very High" complexity, so 1.66 * 1.075 * 1.3 = 2.319850 rounds to 2.32 .

Question 8. AI and Code Synthesis (15 points)

Answer the following questions about AI and code synthesis.

(a) (5 points)

A hypothetical new French software engineer wants to use generative AI to write a procedure. She would like a procedure that drops the fourth element of a list. In French, this might be phrased as "déposer le quatrième".

Here is an actual interaction with ChatGPT 5.0:

- Prompt: Can you write me a small Python procedure deposerquat() that accepts a list X of numbers and returns a list Y. Here are some example tests it must comply with: deposerquat([1,2,3,3]) = [1,2,3] and deposerquat([8,6,7,7]) = [8,6,7] .

  ChatGPT: It looks like `deposerquat()` should return the input list with any trailing run of duplicates collapsed to a single value.

```
[1,2,3,3] → [1,2,3] (last two 3’s become one 3)
[8,6,7,7] → [8,6,7] (last two 7’s become one 7)
```

Here is a simple Python procedure that does exactly that:

```
def deposerquat(X):
    if not X:
        return []
    Y = X[:]  # copy
    # Remove duplicates at the *end* until the last two differ
    while len(Y) > 1 and Y[-1] == Y[-2]:
        Y.pop()
    return Y
```

minutes remaining
Hide Time
Manual Save
Navigation
- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

Unfortunately, the result does not match her goals, even though she used formal tests to communicate her intention.

1. One friend tells her that she should provide more and better tests, such as by using the test input generation approaches in Homework #2 (https://eecs481.org/hw2.html).
2. Another friend tells her that she should use code inspection and carefully read the provided code (as in https://eecs481.org/lectures/se-07-codereview.pdf).
3. A third friend suggests that she should focus on requirements elicitation techniques to avoid terminology conflicts (as in https://eecs481.org/lectures/se-14-valid.pdf).
4. A fourth friend suggests that she use symbolic program synthesis techniques that are not based on large language models (as in https://eecs481.org/lectures/se-24-repair.pdf or https://eecs481.org/lectures/se-26-ai-1.pdf).

For each one of these four proposed approaches, give a reason why it is unlikely to work (i.e., could reasonably fail) for a new software developer. Use at most eight sentences.

Your answer here.

ANSWER:
Homework #2 methods of automated test input generation are not applicable here because they require you to already have the source code! Answers suggesting that not novices may not know what tests to generate are mistakenly addressing HW #1 rather than HW #2. Students may also point out that HW #2 methods like AFL still suffer from the Oracle problem and require you to konw what the right answer is.

Code inspection may fail because it is slow and labor intensive (e.g., a few hundred lines per hour). Novice developers may be unwilling to do it in practice because Generative AI is supposed to save time. Answers suggesting that novices may not know what to look for are partially correct at best: slide 53 of https://eecs481.org/lectures/se-07-codereview.pdf gives an example of Checklists that novices can follow even if they are not familiar with the domain, and code comprehension can still benefit from beacons and indenting and other elements that are common in AI-generated code.

RE Terminology techniques may fail because you have to iterate on known-confusing terminology. Techniques like building a glossary (slide 31 of https://eecs481.org/lectures/se-14-valid.pdf) to pass to ChatGPT likely only work if you know which terms to put in it.

Non-LLM symbolic program synthesis suffers from the same potential overfitting problems if the test suite is incomplete. Slide 51 of https://eecs481.org/lectures/se-24-repair.pdf gives an example with sorting.

1. +1 Explains why HW2-style automated test generation could reasonably fail for this scenario (e.g., requires the code to already exist; does not solve terminology mismatch; novices may not generate the right test oracles).
2. +1 Explains why code inspection could reasonably fail for a novice (e.g., slow, labor-intensive, unlikely to be used in practice; AI-generated code may reduce motivation or capacity to inspect thoroughly).
3. +1 Explains why requirements elicitation / terminology clarification could reasonably fail (e.g., novices may not know which terms are ambiguous; glossaries rely on correctly anticipating the conflict).
4. +1 Explains why symbolic/non-LLM program synthesis could reasonably fail (e.g., still depends on complete test suites; an overfit incomplete specs just like LLMs; may require expertise to use effectively).
5. +1 Writing respects the constraints (≤ eight sentences, coherent reasoning for all four approaches).
6. 0 overall if it appears to be unrelated to class materials (i.e., fabricated)

(b) (5 points)

Consider Endres et al.’s *Can Large Language Models Transform Natural Language Intent into Formal Method Postconditions?* (https://people.cs.umass.edu/~mendres/papers/CanPostconditions.pdf). Research Question 1 in that Microsoft report concludes the following about their `nl2postcond` technique:

- LLMs are good at producing correct postconditions from informal natural language specifications. All prompt variants

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

> generate a correct postcondition for at least 77% and up to 96% of problems. …

In HW2, you used **Randoop** to generate postconditions automatically (https://eecs481.org/hw2.html).

On the surface, these may seem like very similar tools. They both reduce software maintenance costs by making postconditions automatically. In practice, what they require of you may be more important than slight differences in their numerical accuracies — especially since those accuracies may vary from year to year as new models and improvements are developed. Address each of the following 8 questions:

1. Does Randoop need to see your source code to operate or to obtain good results? (yes/no)
2. Does nl2postcond need to see your source code to operate or to obtain good results? (yes/no)
3. Does Randoop normally run your code? (yes/no)
4. Does nl2postcond normally run your code? (yes/no)
5. Does Randoop need prose text from you describing desired behavior? (yes/no)
6. Does nl2postcond need prose text from you describing desired behavior? (yes/no)
7. How does Randoop determine if a generated test is positive or negative? (1-2 sentences)
8. How does nl2postcond determine if a postcondition is correct or not? (1-2 sentences)

(Answer all eight in order in the same text box.)

Your answer here.

ANSWER:

	Randoop	nl2postcond
Need To See Source	yes	no
Runs Your Code	yes	no
Needs Prose Text	no	yes

For a Randoop test, Randoop runs the code dynamically to see if it throws an exception. If it throws an exception it is a negative error test, and if it does not it is a positive regresttion test.

For nl2postcond: “Test-set correctness. … Henceforth, we may refer to "test-set-correct" as simply correct, since correctness in the remainder of the paper is with respect to the provided tests”. That is, they run the whole test suite against the code with the postcondition. If the postcondition is True every time, it is deemed correct (or test-set-correct).

1. Up to +1.5 for the six yes/no answers (each worth +0.25):
    - +0.25 Randoop needs source code → yes
    - +0.25 nl2postcond needs source code → no
    - +0.25 Randoop runs your code → yes
    - +0.25 nl2postcond runs your code → no
    - +0.25 Randoop needs prose text → no
    - +0.25 nl2postcond needs prose text → yes
2. +1 Correctly explains how Randoop distinguishes positive vs. negative tests (must mention running the code and exceptions determining failure).
3. +1 Correctly explains how nl2postcond determines correctness (must mention test-set correctness: postcondition must hold for all tests).
4. +1 Presents all eight answers clearly and in the correct order .
5. +0.5 Writing is concise and respects the prompt constraints (1–2 sentences for each explanation).
6. -1 per any mention of behavior or claim that is fabricated or contradicts the readings.

Extra Credit

(1) Name a course staff member (Derek, Hanchi, Jesse, Priscila, Rohit, Sathvika, or Wes) and list something that person did that helped you learn. (1 point)

Your answer here.

(2) What is your least favorite part of the class so far? Alternatively, what improvement would you suggest for future semesters? (1 point)

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Question 8
- Extra Credit
- Pledge & Submit

Your answer here.

(3) Choose any optional reading (it must say `Optional:` before it on the Lectures page) or any post tagged `long-instructor-post` on Piazza that was not about course logistics (i.e., "Willpower ..." or "Persuading those in Power" or "Hiring, leetcode ..." would count, but "Exam 1, ChatGPT ..." or "Course Expectations ..." would not). Identify it and demonstrate to us that you have read it critically by relating it to your own personal experience, going *beyond* a Generative AI Summary. (We may not give credit if your answer looks like an impersonal AI summary.) (2 points)

Your answer here.

(4) Choose any other optional reading (it must say `Optional:` before it on the Lectures page) or any post tagged `long-instructor-post` on Piazza that was not about course logistics (i.e., "Willpower ..." or "Persuading those in Power" or "Hiring, leetcode ..." would count, but "Exam 1, ChatGPT ..." or "Course Expectations ..." would not). Identify it and demonstrate to us that you have read it critically by relating it to your own personal experience, going *beyond* a Generative AI Summary. (We may not give credit if your answer looks like an impersonal AI summary.) (2 points)

Your answer here.

(5) Which Generative AI tool(s) and version(s) did you use on this exam, if any? For which parts of the exam did you find them (or speculate that they would be) helpful or unhelpful? (Remember, free GenAI is allowed, so this is not cheating. This is to help us improve the course, not to get you in trouble.) If you used any other sources that you are uncertain about, you can cite them here. (1 points)

Your answer here.

Honor Pledge and Exam Submission

You must check the boxes below before you can submit your exam.

☐ I have neither given nor received unauthorized aid on this exam.

☐ *I am ready to submit my exam.*

Submit My Exam

Once you submit, you will be able to leave the page without issue. Please don't try to mash the button.

The exam is graded out of 100 points.