Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

Question 1. Word Bank Matching (1 point each, 14 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. — Creational Design Pattern	B. — Delta Debugging	C. — Elicitation	D. — Fault Localization
E. — Informal Goal	F. — Maintainability	G. — Medical Imaging	H. — Priority
I. — Productivity	J. — Profiling	K. — Program Synthesis	L. — Quantum Computing
M. — Requirements	N. — Risk	O. — Stakeholder	P. — Static Analysis
Q. — Structural Design Pattern	R. — Validation	S. — Watchpoint	T. — Weak Conflict

Q1.1: M

You are called in by metamorphic rock shop company AllSlate to create a website for them. They tell you the things that the website needs: a feature to search through their rock catalog, a feature to add rocks to a user's cart, and the ability to purchase the rocks in their cart securely.

Q1.2: N

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

Q1.3: H

Internally, developers find a defect related to the security of customer records that could have significant consequences if exploited. However, CreditCo management is worried about market reactions and decides not to publicly pursue creating and deploying a patch in the short term.

Q1.4: R

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

Q1.5: P

Maya is facing a looming deadline and is tasked with identifying methods that are likely to be slow. Rather than running the program and recording execution times, she writes a script to count the number of loops in each method's source code and uses that to estimate how long it will take that method to run.

Q1.6: I

Decades ago, software engineering work was mistakenly believed to be partitionable. In practice, adding more people to a late project tends to make it later, informing our ability to plan with respect to *this concept*.

Q1.7: T

ToSoftware is working on a difficult game. Developers are instructed to be sure that "no one can complete the game in under 5 hours" but also that "speedrunners can complete the game in times that can make high-revenue YouTube videos".

Q1.8: Q

Gabriel is writing a program that has access to video files and he wants to use an analysis library that operates on sets of images. He writes a wrapper extracts the frames from the videos as individual images and invokes the library on those individual images.

Q1.9: S

Yang is responsible for maintaining a circularly-linked list. Unfortunately, on some long runs the outgoing pointer from the first element is replaced with an incorrect value. Yang runs the program and arranges for execution to pause when that pointer value is changed.

Q1.10: E

Golda 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 Golda to help guide software development.

Q1.11: K

In an approach conceptually similar to machine learning, Pedro carries out a task manually and then uses a formal grammar to automatically create code that does that same task.

Q1.12: A

Li Bai structures his class so that the normal constructor cannot be called and another method must be called instead. This allows him to hide type information.

Q1.13: O

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

Q1.14: L

Wei's manager mistakenly believes that this technology will definitely allow the hardest problems in computing to be solved in polynomial time.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

Question 2. Delta Debugging (20 points)

We want to design a new fault localization algorithm, Tetra Debugging, which functions very similar to delta debugging but hopefully converges on an answer more quickly. Rather than splitting the test set in half, Tetra Debugging partitions the test set into fourths and tests if each subset is interesting. A brief implementation of parts of the `tetra` algorithm is provided below.

```
1  def split(lst):
2      pivot = math.ceil(len(lst) / 4)
3      return lst[:pivot], lst[pivot:pivot*2], lst[pivot*2:pivot*3], lst[pivot*3:]
4
5  # def minimize_lt_four(lst):
6      # Returns a list of the minimal interesting subset of `lst`
7      # Assumes that lst is interesting and len(lst) < 4
8      # Makes exactly one call to Interesting() for each element of `lst`
9      # The code is not shown, but you can assume it works correctly
10
11 # def union(*lsts):
12     # Returns the union of the lists passed in as arguments
13     # The code is not shown, but you can assume it works correctly
14
15 def tetra(P, C):
16     if len(C) < 4:
17         return minimize_lt_four(C)
18     p1, p2, p3, p4 = split(C)
19     if (interesting(p1)): return tetra(P, p1)
20     if (interesting(p2)): return tetra(P, p2)
21     if (interesting(p3)): return tetra(P, p3)
22     if (interesting(p4)): return tetra(P, p4)
23
24     result = union(
25       tetra(union(P, p2, p3, p4), p1),
26       tetra(union(P, p1, p3, p4), p2),
27       tetra(union(P, p1, p2, p4), p3),
28       tetra(union(P, p1, p2, p3), p4))
29
30     return result
31
```

(a) (4 points) Pick the code snippet that, when put in place of lines 24-28, makes `tetra` yield a correct minimal subset. If there are multiple correct answers, pick any one of them. (Assume there are no syntax errors or similar concerns: this is a question about algorithm logic, not Python details.)

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit


```
result = union(
    tetra(union(P, p1, p2, p3, p4), p1),
    tetra(union(P, p1, p2, p3, p4), p2),
    tetra(union(P, p1, p2, p3, p4), p3),
    tetra(union(P, p1, p2, p3, p4), p4)
)

```

```
result = union(
    tetra(union(P, p3, p4), union(p1, p2)),
    tetra(union(P, p1, p2), union(p3, p4))
)

```

```
result = union(
    tetra(union(P, p2, p4), p1),
    tetra(union(P, p2, p4), p3),
    tetra(union(P, p1, p3), p2),
    tetra(union(P, p1, p3), p4)
)

```

```
result = union(
    tetra(P, p1),
    tetra(P, p2),
    tetra(P, p3),
    tetra(P, p4)
)

```

```
result = union(
    tetra(union(P, p3, p4), p1),
    tetra(union(P, p3, p4), p2),
    tetra(union(P, p1, p2), p3),
    tetra(union(P, p1, p2), p4)
)

```

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

ANSWER:
This question tests the student's knowledge of how delta debugging handles interferference between the two halves of the test set. In delta debugging, the algorithm finds the minimal subset of the first half that, when unioned with P and the second half, is interesting. It then finds the minimal subset of the second half that, when unioned with P and the first half, is interesting.

In tetra debugging (and also an algorithm where we split the test set into three, five, or any other number of subsets), we want to find the smallest subset of each of these smaller test sets that is interesting when unioned with P and the other smaller test sets. If one of the smaller test sets is interesting without intereference, then you can rule out the other three smaller test sets. Otherwise, you must resolve interference between the smaller test sets. In the "interference resolution" part of delta debugging, which this questions tests, any recursive call that `tetra(P, C)` makes to `tetra(p, c)` must satisfy the following: `union(p, c) == union(P, C) and len(intersection(p, c)) = 0`.

As many students noticed, this problem was made more difficult by the fact that the provided implementation of `tetra` has a bug that would be fixed if any call to `interesting(x)` were replaced with `interesting(union(P, x))`. Because of any confusion that this bug may have caused, we will be accepting for full credit any of the answers that resolve interference in some cases but may not in some edge cases. The answers that do not resolve interference will be marked incorrect.

`result = union( tetra(union(P, p3, p4), union(p1, p2)), tetra(union(P, p1, p2), union(p3, p4)) )` is the correct answer and will receive full points.

`result = union( tetra(union(P, p3, p4), p1), tetra(union(P, p3, p4), p2), tetra(union(P, p1, p2), p3), tetra(union(P, p1, p2), p4) )` resolves interference in many cases but misses intereference between `(p1, p2)` and also misses interference between `(p3, p4)`. It will receive full points.

`result = union( tetra(union(P, p2, p4), p1), tetra(union(P, p2, p4), p3), tetra(union(P, p1, p3), p2), tetra(union(P, p1, p3), p4) )` resolves interference in many cases but misses intereference between `(p1, p3)` and also misses interference between `(p2, p4)`. It will receive full points.

`result = union( tetra(union(P, p1, p2, p3, p4), p1), tetra(union(P, p1, p2, p3, p4), p2), tetra(union(P, p1, p2, p3, p4), p3), tetra(union(P, p1, p2, p3, p4), p4) )` does not resolve interference in any case, as `tetra(union(P, p1, p2, p3, p4), p1)` will almost always return the empty set. It will not receive points.

`result = union( tetra(P, p1), tetra(P, p2), tetra(P, p3), tetra(P, p4) )` does not resolve interference in any case, as `tetra(P, p1)` will almost always return `p1`. It will not receive points.

(b) (2 points each, 6 points) Consider an `Interesting` function that returns true if a list contains the elements `2` and `7`. When **Tetra Debugging** is run on the following inputs, how many probes to `Interesting` are made to obtain the minimal interesting subset? (When answering, assume Tetra works according to its specification, regardless of your answer above.) If the program does not terminate, returns a subset that is not interesting, or returns a subset that is not minimal, answer "INVALID".

(i) (2 points) `P = [], C = [2, 0, 4, 6, 3, 5, 1, 7]`

Your answer here.

ANSWER: Because of the mistake in the implementation of `tetra`, we will be accepting the answer for a correct implementation of `tetra` and the answer for the spec implementation of `tetra`.
correct implementation: `10`
spec implementation: `INVALID`

(ii) (2 points) `P = [], C = [1, 6, 3, 7, 2, 4, 5, 0]`

Your answer here.

ANSWER: Because of the mistake in the implementation of `tetra`, we will be accepting the answer for a correct implementation of `tetra` and the answer for the spec implementation of `tetra`.
correct implementation: `10`
spec implementation: `INVALID`

(iii) (2 points) `P = [], C = [2, 7, 6, 1, 5, 4, 0, 3]`

Your answer here.

ANSWER: Because of the mistake in the implementation of `tetra`, we will be accepting the answer for a correct implementation of `tetra` and the answer for the spec implementation of `tetra`.
correct implementation: `3`
spec implementation: `3`

(c) (2 points each, 4 points) Assuming constant time per `Interesting()` check:

(ci) (2 points) What is the worst-case Big-Oh asymptotic time complexity of Tetra with respect to the size **n** of the initial input changeset if a single change induces the failure? (In other words: using Big-Oh notation, how many calls does Tetra make to `Interesting()` if there is one single-element list that is interesting?)

Your answer here.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

ANSWER: Tetra debugging has the same asymptotic performance characteristics of Delta debugging, but with slightly different constant factors. If a single change induces the failure, both Delta and Tetra debugging are the same as binary search, which is `O(log n)`.

(cii) (2 points) What is the worst-case Big-Oh asymptotic time complexity of Tetra with respect to the size **n** of the initial input changeset?

Your answer here.

ANSWER: Tetra debugging has the same asymptotic performance characteristics of Delta debugging, but with slightly different constant factors. The worst case time complexity of both Delta and Tetra debugging is `O(n)`.

(d) (1 point each, 6 points) Consider the following situations. For each situation, indicate whether **Delta Debugging** will obtain a minimal interesting subset and whether **Tetra Debugging** will obtain a minimal interesting subset.

ANSWER: Because Tetra debugging has the same assumptions as Delta debugging (unambiguity, monotonicity, and consistency), the situations in which Delta debugging and Tetra debugging are appropriate to invoke are the same.

Situation #1: The day before project 5 was due, your teammate committed 8 changes to your project repository, collectively causing your code to fail all of your test cases. That night, they pushed 8 more changes, one of which fixed the bug and made the code pass all of the test cases. From the full set of 16 changes your partner committed, you want to find the original change that caused the test cases to fail.

(i) (1 points) **True / False**: Delta debugging can obtain a minimal interesting subset for situation #1.

◯ True
◯ False
ANSWER: False

(ii) (1 points) **True / False**: Tetra debugging can obtain a minimal interesting subset for situation #1.

◯ True
◯ False
ANSWER: False

ANSWER: This example is not an appropriate use of delta/tetra debugging because the interesting function is not monotonic. Consider a set of just the change that introduces the bug which causes the test cases to fail; this set is interesting. Now, consider a set which consists of both the change that introduces the bug and the change that fixes the bug. This set is not interesting, even though it is a superset of an interesting set.

Situation #2: We have an HTML tag `<div class="widget" role="navigatino" aria-lable="Outline" tabindex="0" style="padding-right: 16px;">` that fails to parse with our HTML parser. We want to find which tag attribute may be causing the tag to fail to parse.

(iii) (1 points) **True / False**: Delta debugging can obtain a minimal interesting subset for situation #2.

◯ True
◯ False
ANSWER: True

(iv) (1 points) **True / False**: Tetra debugging can obtain a minimal interesting subset for situation #2.

◯ True
◯ False
ANSWER: True

ANSWER: This is an appropriate use case for delta/tetra debugging. There is a single unique attribute which is causing the tag to fail to parse, so unambiguity holds. Either the tag parses successfully or fails to parse, so consistency holds. If a single attribute fails to parse then the whole tag fails to parse, so monotonicity holds as well.

Situation #3: We are putting together a software development team, and among our whole team, we need at least one developer who knows each of the languages Python, C++, and C#. We have the following list of candidates: { Aidan: [Python, C#], Cameron: [Python], Conner: [C#], Daniel: [C++], Holly: [C++, C#], Jason: [Python, C++, C#]} and want to find the smallest team we can assemble that together knows all of the languages.

(v) (1 points) **True / False**: Delta debugging can obtain a minimal interesting subset for situation #3.

◯ True
◯ False
ANSWER: False

(vi) (1 points) **True / False**: Tetra debugging can obtain a minimal interesting subset for situation #3.

◯ True
◯ False

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

ANSWER: False

ANSWER: This is an inappropriate use case for delta/tetra debugging because it does not satisfy the unambiguity assumption. Consider the teams [Aidan, Daniel] and [Cameron, Holly]. Both teams are interesting because at least one team member on each team knows Python, C++, and C#. However, the intersection of the two teams, [], is not interesting because it does not have members that know all of the languages.

Question 3. Short Answer (4 points each, 20 points)

(a) (4 points) Lizzy is making preparations to open her new hot air balloon resale shop on State Street called *Maize and Blue'ns*. Her store will sell locally to Michigan but also online to Ohio. The local and online sales have different taxes and policies. In 3 sentences or fewer, describe which design pattern you would use and why, and one risk associated with that design pattern.

Your answer here.

ANSWER: Multiple answers are possible. However, the question intentionally favors the Template Method. The Strategy design pattern is also a decent fit, but is slightly harder to support. In class (e.g., slide 35 of the Design for Maintainability lecture) we explicitly considered the situation of a store that sells in different states with different tax policies. Students could argue for Strategy (which is extensible and separates the algorithm from the client), but the Template Method is likely a better fit since there are explicitly invariant parts (e.g., doing sales, tracking stock, etc.) and changing parts (different states with different policies). Students could lose points by confusing multiple methods (e.g., saying Template Method but then listing features associated with Strategy) or by listing Strategy but not providing enough support.

(b) (4 points) Cassie is adding a new feature to decrease wait times at her carnival attraction Euphoric Carousel. Describe four steps or activities she might follow for effective requirement elicitation. Use 4 sentences or fewer.

Your answer here.

ANSWER: This question admits a significant amount of freedom. Students could describe 'capturing and representing knowledge', 'identifying stakeholders', 'understanding the domain', 'interviews', and so on. Students could also list activities that make a high-level step effective, such as 'ask follow-up questions' or 'begin with specific questions' for effective interviews. Students could also list conflict resolution activities such as 'build a glossary' or 'explore tradeoffs'.

(c) (4 points) Describe 2 considerations Ray Buse mentioned during his lecture regarding how Google automates the testing of phone applications. For each consideration, describe the problem and then describe the impact or solution. Use a total of 4 sentences or fewer.

Your answer here.

ANSWER: This question required students to list something from the Buse lecture in particular, so flexibility was limited. The list below is fairly exhaustive. In addition, the question focused on *how* (and not *why*) Google automates application testing.

1. Recognize that not all screens are equal, but not all are unequal
2. Recognize that all phones behave differently when interacting with the app — best to use physical devices vs. emulators
3. Identify target goals to determine which abstractions to take (supervised learning)
4. Use unsupervised learning — data forms clusters and we can determine from there which is important — still learning how to get better at this
5. Mitigate the problems associated with setting up a device lab (device overheating, cleanup, radio/electromagnetic frequency interference, power supply issues)
6. Consider social responsibility — needs to protect users from malware, apps must be accessible for those with impairments or disabilities
7. Test new versions of Android's interactions by focusing on top apps

8. Find and deal with disruptive ads
9. Figure out how a new feature integrates with existing code and user flow

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

(d) (4 points) List two similarities between Program Synthesis and Pair Programming. Then list two differences. Strong answers should include a focus on the inputs and outcomes of the two tools — on when they can be used and what benefits they provide. For example, do not simply say that one involves a second human and the other does not. Instead, compare and contrast them in terms of their potential use in a software development process. Use at most 4 sentences.

Your answer here.

ANSWER: Program Synthesis and Pair Programming both involve one agent producing code under the guidance of another agent. Program Synthesis and Pair Programming both have the potential to reduce overall development effort. Pair Programming favors communication, brainstorming and the generation of ideas and example — Program Synthesis often requires examples or demonstrations to make progress. Pair Programming and Program Synthesis can both be useful when creating or prototyping new code.

However, Pair Programming is likely to be useful in debugging or fault localization activities, but Program Synthesis is mostly used only in code creation. Pair Programming may take more time than programming alone (even if it is a net advantage due to, for example, reduced defects), while Program Synthesis (when it can solve a problem) is almost always faster than a single programmer. Program Synthesis typically produces code in a restricted domain-specific language while Pair Programming can produce code in any language both participants are familiar with.

A full-credit answer should touch on the notion that the typical benefits of Pair Programming may include smaller code, lower defect rates, and happier programmers ... but not necessarily reduced time. By contrast, Program Synthesis focuses on reducing programming time (and one could make an argument that Program Synthesis requires many examples or demonstrations to produce correct code, as in some of the FlashFill examples).

(e) (4 points) Describe, in your own words, a project that might require multiple languages. Then, provide one advantage and one disadvantage of multi-language projects. Use at most 4 sentences.

Your answer here.

ANSWER: Projects can vary from things like TensorFlow to personal projects. 1 point for vague answer (e.g., a project that needs a fast C kernel), 2 points for a description of a project that involves multiple languages but no cross-language interaction, and full credit for a project that describes in detail a project where multiple languages interact.

Advantages: Some processes can be more effectively coded or optimized in an alternate language, provides more functionality and flexibility for projects, increasingly common, etc.
Disadvantages: Integrating data and control flow across languages can be difficult, debugging can be harder, building becomes more complicated, developers must have expertise in more than one language.

Question 4. Fault Localization (10 points)

You are hired to write a program that generates random questions for a midterm exam. For each input (the name of a user taking the exam) the program should generate slightly different exam text as its output. You write a long Python program to do this, but unfortunately the output of your program is sometimes incorrect. In other words, for some inputs your program produces the correct output but for some other inputs your program produces incorrect output.

```
1   def quality_question():
2     random.seed(int(username))
3     username = argv[2]
...
17    prompts = [alternate_1, alternate_2, alternate_3]
...
37    # Randomly sample two prompts
38    a_prompt, b_prompt = random.sample(prompts, 2)
...
89    c_prompt = generate_prompt()
90    # If the randomly generated prompt is already chosen, regenerate it
91    if c_prompt in [a_prompt, b_prompt]:
92      c_prompt = generate_prompt()
...
```

```
150 def name_scrambler():
...
```

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

You decide to use fault localization to pin down where the issues in your code are.

(a) (4 points) You consider two automated approaches to fault localization: an approach in which Delta Debugging is used to find a minimal set of suspicious lines, and an approach in which Tarantula is used to rank suspicious lines. (The details of these two potential approaches are intentionally not provided. You must think of possibilities based on course concepts.) Which fault localization method would work better here: Delta Debugging or Tarantula? Why? (Use at most four sentences.)

Your answer here.

ANSWER: Typically Tarantula. Tarantula is explicitly designed to rank suspicious lines based on observed coverage information. In the problem setup, it suggests that you know some inputs for which your program works correctly and some for which it does not: this is exactly the setup that Tarantula favors. By contrast, DD finds a minimal interesting set. While it is tempting to imagine a notion of Interesting corresponding to 'is part of the fault', it's not clear how to do that. How would Interesting tell if lines are suspicious or not? A full-credit answer would clarify not just that Tarantula is a good choice but that DD is a poor one.

A partial credit answer might be given for DD. In class we discussed how Tarantula-style algorithms often fail when fault localization depends on the value of a string (e.g., for Cross-Site Scripting or SQL Code Injection) and this problem is set up so that the input values (usernames) are all strings. However, the strings are just used to initialize random seeds, so this would be a difficult argument to support fully.

(b) (4 points) Consider the following table of program runs. Each row corresponds to one test case execution. In other words, each row reports one run of your program in which it takes a username as input and produces output that is either correct or incorrect. Each row also includes the lines visited while your program executes on that input.

Test Input	Passed?	Lines visited
bakalm	True	[1, 2, 3, 17, 37, 38, 89, 90, 91, 150, 151]
weimerw	True	[1, 2, 3, 17, 37, 38, 89, 90, 91, 92, 150, 151]
hstauff	False	[1, 2, 3, 17, 37, 38, 89, 90, 91, 92, 150]
chein	False	[1, 17, 37, 38, 89, 90, 91, 92, 150]
haasea	True	[1, 2, 3, 17, 37, 38, 89, 90, 91, 150]

Compute the **Tarantula** suspiciousness score for each line in the program and provide the suspiciousness scores for the top 3 most suspicious lines of code. Express the final answer as a list of tuples of line numbers (ints) and scores (floats, 3 significant figures), sorted by score descending and then by line number ascending; if line 2 has a suspiciousness of 0.667, line 1 has a suspiciousness of 0.5, and line 5 has a suspiciousness of 0.5, your answer should be `[(2, 0.667), (1, 0.500), (5, 0.500)]`.

Your answer here.

ANSWER: [(92, 0.75), (1, 0.5), (37, 0.5)]

(c) (2 points) Which line from your list do you think is causing the problem and why? (If you think your list does not contain a relevant line, indicate that instead and explain why.)

Your answer here.

ANSWER: 92 is the most suspicious with a suspiciousness of 0.75. The answer should be one of the most suspicious, but the question is pretty holistic. Credit will be given for any answer that provides a line that is likely causing the failure.

Question 5. Profiling (7 points)

Consider the function call profile below. The `main` function is run with an *event-based profiler* and the following hierarchical profile is generated:

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit


```
1 * main()
    2 * can()
        2 * head()
        3 * kay()
        4 * ever()
    1 * direction()
        5 * head()
        5 * kay()
        5 * ever()
```

In addition to the call graph above, the following non-cumulative, per-one-single-call *self time* information is also generated:

```
main() - 100ms
can() - 250ms
direction() - 720ms
head() - 50ms
kay() - 100ms
ever() - 40ms
```

Note, the call graph specifies how many times a function is called in its individual context. For example, each time `can` is called, `head` will run multiple times.

(a) (5 points) In this problem, the self-time of a function is the time the function's stack frame spends on the top of the stack for one single call to that function. (Equivalently, the self-time of a function is the time taken by one single run of that function *not* including the time taken by any children or parent functions.)

`main` is run with a statistical profiler. Assuming that the self-times remain constant across runs, write the names of each function in descending order of probability that a random probe of the profiler would interrupt the program in that function. If there is a tie, list the function with the alphabetically-earlier name first (e.g., so if you believe "ant" and "bat" are equally likely to be running when the program is sampled, list "ant" before "bat"). Your answer must be a Python-formatted list. For example, if you believe that the order should be (most probable) A, B, C, D (least probable), you should answer: `["A", "B", "C", "D"]`. Your list should contain 6 elements.

Your answer here.

ANSWER: `['kay', 'direction', 'ever', 'can', 'head', 'main']`

`{'kay': 1100, 'direction': 720, 'ever': 520, 'can': 500, 'head': 450, 'main': 100}`

(b) (2 points) Support or refute the claim that a call-graph execution profiler like `gprof` would produce actionable insights (i.e., useful information) for a project that includes both Java and C code. Use two pieces of evidence to support or refute the claim. Use at most 4 sentences.

Your answer here.

ANSWER: Likely refute. Execution profilers typically focus on a particular language (such as C or Python or Java). The Multi-Language Projects lecture mentioned a number of challenges associated with using dynamic analysis tools or profiles across languages. In a standard setting, where Java code is interpreted by a Java Virtual Machine, a profiler like gprof would not be able to see the names of the methods executing much less their call graphs. gprof looks at the 'machine stack' (e.g., the value of the program counter), not the 'Java Virtual Machine stack'. A full-credit answer should mention relevant evidence (e.g., what sampling-based profiling needs to work, what stack information a call-graph profiler needs to work, how a multi-language Java/C project works, etc.).

Question 6. Interviews (14 points)

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

As an interviewer, you give the following technical challenge to a potential candidate: "Write `isPalindrome()`, a function that returns `true` if parameter `x` is a palindrome integer. Note that an integer, like 12321, is a palindrome if it reads the same forwards and backwards."

The candidate's implementation of `isPalindrome()` is below along with two questions the candidate asked you:

```
// Q: Can integer x be in range [0-9]? A: Yes.
// Q: Should I account for integer overflow? A: Yes.

bool isPalindrome(int x) {
    // a single digit is a palindrome
    if (x < 10) {
        return true;
    }

  // if x's last digit is 0, then its first digit must be 0 in order
  // to be a palindrome. In this case, only 0 can be a palindrome.
  if (x % 10 == 0 && x != 0) {
    return false;
  }

  // revert the last half of x for comparison against first half
  int revertedNumber = 0;
  while(x > revertedNumber) {
    revertedNumber = revertedNumber * 10 + x % 10;
    x /= 10;
  }

  return x == revertedNumber || x == revertedNumber / 10;
}

```

(a) (2 points) Identify two test cases where the provided `isPalindrome()` would return `false`.

Your answer here.

ANSWER: Answers will vary. Student solutions must not be a palindrome. `{{123}}`, `{{10}}` are potential answers that return false.

(b) (4 points) Identify four things that the candidate did well. (In other words, identify four properties that a company might desire in a software engineer that could potentially be shown by a candidate taking the interview and that were shown by this particular candidate.) Use at most 4 sentences.

Your answer here.

ANSWER: Answers will vary. Potential solutions include the following:

1. The candidate provided inline comments explaining some of their code.
2. The candidate asked relevant questions regarding code functionality.
3. The candidate has consistent indentation.
4. The candidate used a descriptive variable name.

(c) (4 points) Support or refute the claim that the candidate's implementation of `isPalindrome()` is functionally correct. Use at most 4 sentences.

Your answer here.

ANSWER: Refute. The candidate's implementation of `isPalindrome()` is not functionally correct. The candidate missed a base case regarding `x` being a negative number. Negative integers would not count as a palindrome. `{-313}` reversed is `{313-}`.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

(d) (2 points) Describe a hypothetical defect in `isPalindrome()` that Automated Program Repair would likely be able to fix and explain why. Then describe a hypothetical defect in `isPalindrome()` that APR would be unlikely to fix and explain why. Use at most 4 sentences.

Your answer here.

ANSWER: Automated Program Repair works best in situations where fault localization can narrow down the bug to a few lines, where mutation operators can fix the bug, and where test cases can reveal that the bug is fixed.

An example of a hypothetical bug that would likely be easy to fix would be mistakenly writing `return false;` on line 7 (instead of returning true). That line would be quickly implicated by fault localization, mutation operators can easily flip false to true, and tests will reveal that the fix is correct.

An example of a hypothetical bug that APR would be unlikely to find would be mistakenly writing `return false;` on line 23 (instead of the complicated return logic). The line would be tricky to locate (e.g., it is visited on the same runs that visit the while loop, etc.), but more importantly, mutation operators are unlikely to create that complicated logical "from scratch", even if two or three mutations are allowed.

(e) (2 points) Support or refute the claim that programmer productivity is essentially fixed (i.e., individual performance differences are predominantly due to inborn talents rather than learned skills). Reference at least two results from scientific literature (e.g., from Computer Science or Psychology, etc.) as evidence as you support or refute the claim. Use at most 4 sentences.

Your answer here.

ANSWER: Refute. Especially given the word "predominantly" and the software engineering context, the scientific evidence strongly supports learned skill as more impactful than inborn talent.

- The most direct choice is *Experts bodies, experts minds: How physical and mental training shape the brain* which, in the title, reminds the reader that training (i.e., learned skill) shapes the brain as an explanation for productivity.
- *Decoding the representation of code in the brain* shows that expert brains (as assessed by courses taken, i.e., learned skill) treat programming languages differently.
- *Expertise in Problem Solving* shows that experts (again assessed in terms of school standing, i.e., learned skill) make fewer mistakes, apply rules in a single step ("semantic chunking"), and "cluster" problems differently than do novices.
- *A Human Study of Fault Localization Accuracy* shows fault localization accuracy going up directly with years in a CS major (i.e., learned skill).
- *Talent in the taxi: a model system for exploring expertise* (or "The London Taxi Cab / Bus Driver Study") shows a very direct relationship between time on the job (i.e., learned skill) and changes to the brain (although it does not address productivity directly).

Possible mistakes include choosing resources like *The Economic Value of Rapid Response Time*, which has implications for computer latency but does not address talent vs. skill, or *Exploratory Experimental Studies Comparing Online and Offline Programming Performance*, which does show big differences, but doesn't talk about talent vs. skill, or *The Mythical Man Month* which does not talk about the origins of productivity differences in terms of people.

Question 7. Requirements Elicitation (15 points)

(a) (6 points) Suppose that the university is in the beginning stages of developing a streaming service, specifically for students to watch University of Michigan related videos (sports, news segments, etc.) called BlueTV. BlueTV should have a feature in which it recommends new relevant or related videos to watchers. Name 3 *quality requirements* that the team might have for the video recommender aspect of BlueTV, noting if each is verifiable or informal. Include at least one verifiable requirement and one informal requirement in your answer. Use 3-6 sentences if possible.

Your answer here.

ANSWER: Answers will vary. Students may discuss requirements related to confidentiality, privacy, integrity, availability, reliability, accuracy, or performance (as discussed on slide 37 of the requirements lecture). Verifiable requirements must be able to be tested objectively. For example, a student may say that BlueTV must provide a list of 30 recommended videos within 2 seconds, which would be a verifiable requirement.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit

(b) (6 points) Identify 2 different stakeholders for BlueTV, and describe two different *conflicts* that might arise between the two identified stakeholders during the requirements elicitation process. For each conflict, include a formal keyword to describe the conflict if possible (e.g., what type or name of conflict is it?), and then describe the *strength* of the conflict (if possible). Use at most 6 sentences.

Your answer here.

ANSWER: Answers will vary. Example stakeholders include the University of Michigan sports office, U of M communications, U of M lawyers, and students using BlueTV. One example of a conflict is that the students may want to see the average rating of each video on BlueTV, but a UM official may not want rating information to be displayed to users. This would be a strong conflict. Similarly, there might be a requirement in which "user" refers to students watching videos in one statement and officials uploading videos in another statement: this is a designation clash. Example: "users must not be able to delete videos" (meaning student users) vs. "videos posted by users must be available for viewing within 30 minutes" (meaning official users). This would also be a strong conflict.

(c) (3 points) Choose one of the conflicts you identified previously. What would be the best way to resolve this conflict? Use at most 3 sentences.

Your answer here.

ANSWER: Answers will vary. Refer to slides 30 - 31 of the Requirements Elicitation lecture. One key word to look for is negotiations. The student may also mention some form of prototype, mockup, or storyboard. For example, based on the answer above, a student might suggest the team negotiate with both stakeholders by presenting a prototype with a more generalized "like dislike" rating system. A terminology conflict might be resolved by building a glossary.

Question 8. Extra Credit (1 point each)

(*Feedback*) What was your favorite topic or activity during the course?

What is one thing you like about this class?

(*Feedback*) What do you think we should do more of next semester (or what is the thing you would most recommend that we change for future semesters)?

What is one thing you dislike about this class?

(*Guest Lecture*) List one thing you learned from guest speaker Peter K. Shultz of Microsoft or otherwise convince us that you paid careful attention during that lecture. Your answer must be distinct from any references to the Shultz lecture you may have made earlier in the exam.

Shultz guest lecture.

(*Optional Reading 1*) Identify a single optional reading that was assigned after Exam 1. Write two sentences about it that convince us you read it critically. (The most common student mistakes for these questions in Exam 1 were choosing a required reading instead of an optional reading or failing to "identify" or name the reading selected.)

Optional Reading 1

(*Optional Reading / Piazza 2*) Identify a different single optional reading that was assigned after Exam 1 or a *"long instructor post"* that was posted on Piazza after Exam 1. Write two sentences about it that convince us you read it critically.

Optional Reading 2

(*Guest Lecture*) List one thing you learned from guest speaker Dr. Ray Buse of Google that was not listed on an introductory summary slide or otherwise convince us that you paid careful attention during that lecture. Your answer must be distinct from any references to the Buse lecture you may have made earlier in the exam.

Buse guest lecture

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit