Question 1. Word Bank Matching (1 point each, 14 points)

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

A. — Automated Program Repair	B. — Composite Design Pattern	C. — Concurrency Bug	D. — Conditional Breakpoint
E. — Delegation	F. — Factory method pattern	G. — Fault Localization	H. — Functional Requirement
I. — Informal goal	J. — Interview	K. — Perverse Incentive	L. — Postconditions
M. — Productivity	N. — Profiling	O. — Quality Requirements	P. — Readability
Q. — Requirements Elicitation	R. — Singleton Design Pattern	S. — Stakeholder	T. — Strong Conflict
U. — Top-down comprehension	V. — Validation	W. — Verification	X. — Watchpoint
Y. — Weak Conflict			

Q1.1:

O

Amazoon has created a privacy policy for their latest app, stating that a user's personal information will not be shared with other users without their consent. Their app must implement this policy.

Q1.2:

T

Anipleecs requires all employees to work from home every other day so that desks can be shared,reducing building rental costs. In addition, they just adopted a policy requiring first-year employees to be present in the office five days each week, improving mentoring and engagement.

Q1.3:

Y

Movaghar & Movaghar is developing a new online shopping app and discussing the return policy. Buffy proposes a 90-day limit for product returns, while Gojo disagrees, suggesting that customers should have the flexibility to return products at any time.

Q1.4:

G

Luffy wants to determine which of the 50 patches committed last night cause a bug. To do this, they use delta debugging to find a minimal subset of those patches that induces a failure.

Q1.5:

W

After creating a new application, Wahoo ensures that the application conforms to its specifications.

Q1.6:

P

Leena recently identified a bug in MVidia's banking application that results in users being charged without their money being transferred. To localize the issue, Leena begins reviewing the code base, but finds it challenging. Leena believes that it would be beneficial to improve *this term*.

Q1.7:

S

Miscord is assessing whether selling sports software would be profitable. Their discussions incorporate multiple *this term*, helping to ensure that they consider diverse factors such as legality, software feasibility, company executive opinion, and sales projections.

Q1.8:

Q

Devforce wants to create promotional videos for their debut product. They outsource this project to a third-party company led by Priscila. Priscila first meets with Devforce to discuss the key elements to include in the video.

Q1.9:

U

Yushu is a new employee at GogoWeimer. They utilize their experience from prior jobs to understand the codebase and quickly get up to speed.

Q1.10:

K

123movies introduces a policy offering employees a $25 gift card for every 100 commits they make on GitHub.

Q1.11:

L

Ben is developing a program. GlazeBook specifies that the program should be designed so that the return value is always 0.

Q1.12:

M

Ritij is a recent graduate and new hire at Bank of Michigan. Each week, they complete about one-third of the work completed by a more senior employee.

Q1.13:

A

Mesla Inc. attempts to make maintainers more productive via a system that suggests patches for certain classes of defects, such as null pointer errors.

Q1.14:

I

Boruto is working on an application for EECS Depot. Boruto's manager says that the final product must be 'a secure app.'

Question 2. Guest Lectures and Course Concepts (18.5 points)

The following questions will ask you about the 4 guest lectures. Please read the question carefully and follow instructions as directly as possible.

(a) (3 points)

At least two guest lecturers mentioned their work related to AI for SE. Give one lecturer's name and briefly describe what they did or talked about on this topic in their lecture. (one sentence maximum) Then provide an example of SE for AI that you have learned in EECS 481 class. (one sentence maximum)

ANSWER: Aidan Yang mentioned that he is doing PhD research about using large language models (AI) for program analysis (SE) or to guide verification (SE). (+1.5) Emerson Murphy-Hill mentioned that his current work is about large language model -based developer experiences in Visual Studio, VSCode, and GitHub (e.g., LLMs can help fix GitHub issues). (+1.5) Henry Beckstein's AI for Subaru satellite/computer vision is also a valid answer. (+1.5)

(b) (3 points)

Aidan Yang mentioned four techniques for software quality assurance, starting with software testing. What are the other three techniques he mentioned? (one sentence). EECS 481 discussed model checking and its definition. What technique described by Aidan is the closest to the definition we covered, and why? (two sentences maximum)

ANSWER: Aidan also mentioned property based testing, bounded model checking and non-bounded model checking. (+1.5) In EECS 481, model checking was defined as

algorithmically verifying if a finite-state model of a system satisfies a given specification. The specification is usually expressed in temporal logic. Model checking considers executions of arbitrary length (i.e., an arbitrary number of transitions). This definition is closest to Aidan's explanation of non-bounded model checking, which is model checking and also does not only consider a pre-determined number of execution steps. (+1 non-bounded model checking answer, +1 explanation)

(c) (3.5 points)

Henry Beckstein mentioned software testing at Subaru in his guest lecture. Quote one sentence describing his idea of the purposes of testing from his lecture slides. Then describe, in one sentence, a purpose of software testing from the EECS 481 lectures or readings. To what degree do those two ideas align? Briefly explain your answer. (three sentences maximum)

ANSWER: Henry mentioned in his slides that the purpose of testing is to “Test for functional correctness” / “Test for customer feeling”. (+1) In class, we described the purpose of software testing as software quality assurance: gaining confidence that the implementation adheres to the specification. (+1) The two ideas are essentially the same, because quality consists of functional correctness as well as the (possibly non-functional) satisfaction of the software users, so testing for quality assurance aligns with Henry's description. (+1.5)

(d) (4 points)

Farhad Arbab introduced a programming language in his guest lecture. What kind of program is this new language designed for? (quote one relevant sentence from his slides, and then answer the question in another one sentence) For this kind of program, what is a common issue and what is a solution that we covered in EECS 481 could help resolve that issue? (two sentences maximum)

ANSWER: “There exists a better way to conceive of and express concurrency protocols…”. The language Farhad introduced is for better supporting concurrent programs. (+1 quote, +1 answer) The most common issue associated with concurrent programs is the race condition (when two threads access the same shared state and at least one thread writes to it). An analysis like Eraser can help resolve locking issues on shared variables by reasoning about lock sets. The Delta Debugging lecture also described a use of delta debugging to reason about thread interleavings for concurrent programs. (+2)

(e) (5 points)

Henry Beckstein mentioned software analysis at Subaru in his guest lecture. Quote one sentence from his lecture slides that shows his idea of the use case and limitation of software analysis. In two sentences, describe why this limitation exists for most software analysis techniques. Finally, name two verification approaches from the EECS 481 lectures or readings that might apply in the domain Henry described. (five sentences maximum)

ANSWER: Quote: “Software Analysis can test for memory leaks, but it can’t verify if the program is correct” (+1) Most analysis techniques suffer from false positives and or false negatives (this issue is sometimes called soundness or completeness). Analysis techniques that target interesting software properties rarely make full guarantees about correctness. Such analyses are often undecideable. (+2) Techniques such as model checking (which is often applied to safety-critical domains such as automotive software) or formal code inspection (which can help with tricky bugs like memory leaks in a small piece of code) may apply. (+2)

Question 3. Short Answer (25 points)

You are a manager. You want to figure out how much time your team spends on the following tasks: investigating bug reports, reading requirements, debugging, and browsing Stack Overflow. You have a hypothesis that your team is spending too much time reading bug reports. To assess this, you consider two options: (1) use a software tool that tracks which window each team member has active (i.e., bug report window vs. coding window) and logs how much time they spend doing each activity, including when they switch tasks, or (2) use a software tool that displays a pop-up window to each team member every 15 minutes, asking each person to select the activity they are currently doing from a list.

(a) (4 points)

Identify two profiling-related concepts in the above scenario. For each concept, explain the concept and its relationship to the scenario in two sentences. (2 * 2 = four sentences total)

ANSWER:

+(1) point per correctly-identifyig a profiling concept (1) can de identifed as flat profiler, call-graph, or intrumentation. (2) is sampling profiling

+(1) point per satisfactory explination (paralleling to the above)

Sample Answer: (1) Describes varies types of profiling. Flat profiler, call-graph, and instrumentation are valid here with correct explaination. Instrumentation profiling can makes sense here since we are using a technique that effectively adds instructions to the target

program to collect the required information. Flat and call-graph profilers are both profilers types that are based on output. In this scenario, callling (1) an example of a flat profiler is valid since we are caputuing task "times" but don't care about tracking the context or callees within those tasks. This senario is also similar to a call-graph because we can use the output to determine freqeuences and call-chains. (2) Describes the sampling approach but instead asking a program how long they are taking, we having a human check in every 15 minutes.

You are programming a calendar. When the current month is not December, all events created and added will be regular type events. When the current month is December, all events created and added to the calendar will be of a special holiday type instead. Events can be shown in different fonts. Using some or all of the following method signatures:

```
create_event()

add_event()

is_december()

change_fonts(string font_type)
```

(b) (2 points)

Describe a (bad) way to solve this problem with an anti-pattern in 4 sentences or less.

ANSWER:

+(2) for suggesting reusing if statements throughout the code

OR +(2) for any other antipattern that does not involve the abstract factory design pattern (see second example answer)

Answer: Before calling create event, first check if is_december() is true. If is_december() is true, call holiday_event_factory.create_event(), otherwise call regular_event_factory.create_event(). Then, whichever path was taken, add the resulting event to the calendar using add_event()

Answer: [Alternate prompt] Before calling create event, first check if is_executive_visiting() is true. If is_executive_visiting() is true, call priority_event_factory.create_event(), otherwise call regular_event_factory.create_event(). Then, whichever path was taken, add the resulting event to the calendar using add_event()

Answer: Create two different versions of the calendar program. The first uses regular_event_factory.create_event(), the second uses holiday_event_factory.create_event(). Use the first program during Jan-Nov, use the second program only during Dec.

Answer: [Alternate prompt] Create two different versions of the calendar program. The first uses regular_event_factory.create_event(), the second uses priority_event_factory.create_event(). Use the first program during when no executives are in the office, use the second program only when executives are in the office.

(c) (2 points)

Describe a way to solve this problem with the abstract factory design pattern in 4 sentences or less.

ANSWER:

+(1) mentions using only one conditional statement at the beginning to set a single variable which will be used to create the events

+(1) mentions using an abstract event_factory class and implementing two subclasses for creating regular and holiday events

Answer: Create an abstract factory class called Event_Factory and create two subclasses of Event_Factory, Regular_Event_Factory and Holiday_Event_Factory. Only once at the start of the program, check if it is_december(). If is_december() returns true, assign an Event_Factory variable (event_factory) to be an instance of Holiday_Event_Factory, otherwise set event_factory to be an instance of Regular_Event_Factory. Now, creating and adding events is as simple as calling event_factory.create_event() and add_event().

Answer: [Alternate prompt] Create an abstract factory class called Event_Factory and create two subclasses of Event_Factory, Regular_Event_Factory and Priority_Event_Factory. Only once at the start of the program, check if is_executive_visiting() is true. If is_executive_visiting() returns true, assign an Event_Factory variable (event_factory) to be an instance of Priority_Event_Factory, otherwise set event_factory to be an instance of Regular_Event_Factory. Now, creating and adding events is as simple as calling event_factory.create_event() and add_event().

(d) (2 points)

Give one reason why the abstract factory design pattern is preferred to the anti-pattern in 4 sentences or less. Your answer should identify a desired maintainability property and indicate why the design pattern promotes it.

ANSWER:

+(1) identifies a desired maintainability property

+(1) indicates why the design pattern promotes the desired maintainability property

Answer: (Assuming the student talked about using many conditional statements throughout the code as an antipattern) The abstract factory design pattern requires only a single conditional statement, reducing how many code changes are required if the functionality must change. If the holiday events should instead be created in both December and November, every "if is_december()" statement for events will need to be updated (e.g., "if is_december() or is_november()"). The abstract factory method requires only updating the initial conditional that defines abstract_factory. The abstract factory design pattern promotes this by requiring just a single concrete factory class assignment at the start of the code and putting all of the relevent behavior inside of the different concrete factory subclasses.

Answer: (Assuming the student talked about using many conditional statements throughout the code as an antipattern) The abstract factory design pattern requires only a single conditional statement, reducing how many code changes are required if the functionality must change. If the priority events should instead be created only when executives are in the office on weekdays, every "if is_executive_visiting()" statement for events will need to be updated (e.g., "if iis_executive_visiting() and is_weekday()"). The abstract factory method requires only updating the initial conditional that defines abstract_factory. The abstract factory design pattern promotes this by requiring just a single concrete factory class assignment at the start of the code and putting all of the relevent behavior inside of the different concrete factory subclasses.

(e) (4 points)

You are part of a team of developers that is using model checking for quality assurance. Your model checker assesses whether an abstraction (model) of the program adheres to its formal specification. To directly improve model checking outcomes, one of your teammates proposes expanding the current test suite to obtain higher branch coverage. Support or refute the recommendation in 4 sentences or less.

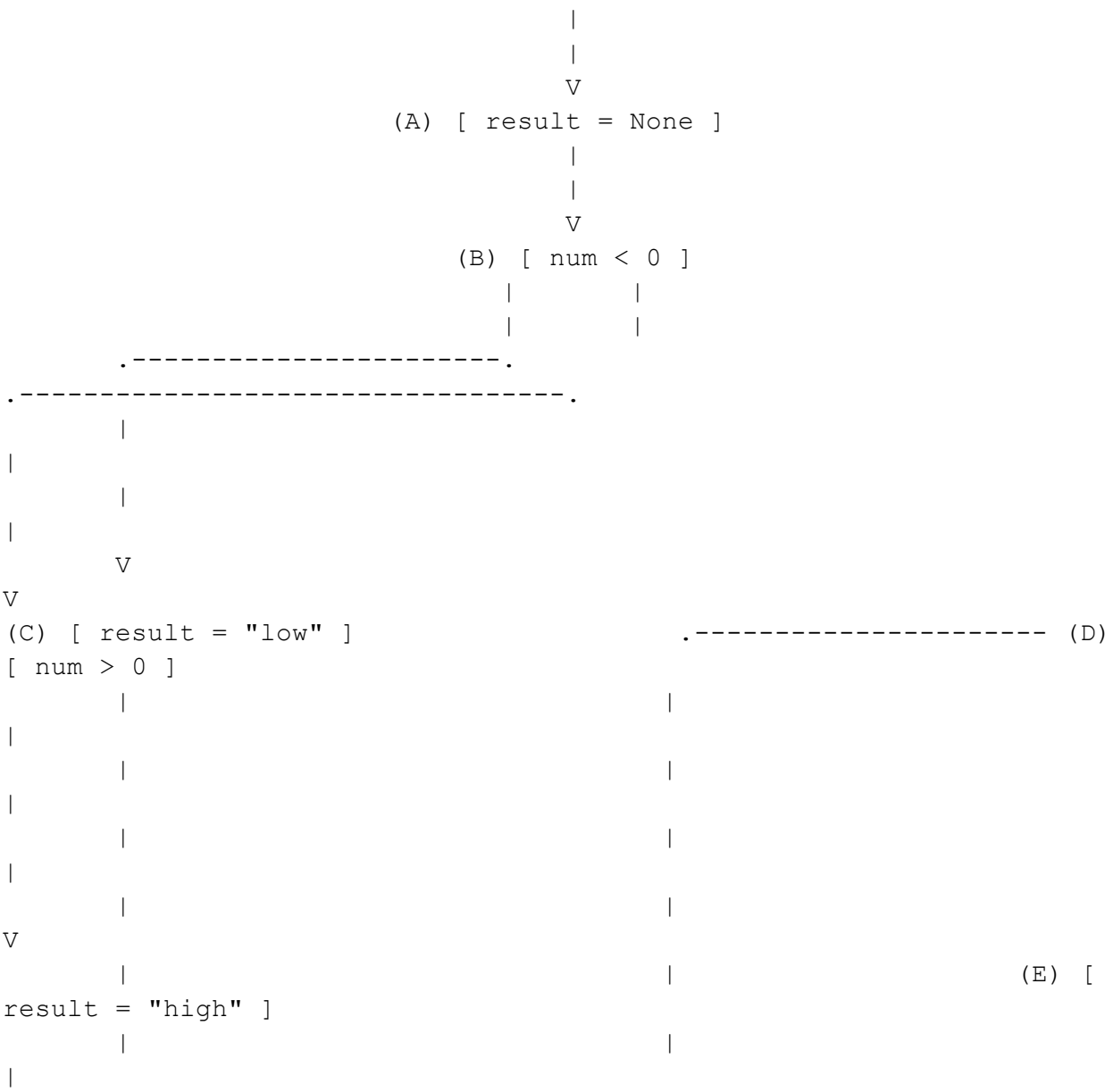
ANSWER:

+2 Refute the reccomedation

+1 Reasoning - Model checking is a static analysis technique that verifies whether a model adheres to a formal specification, and it does not rely on dynamic execution or branch coverage of the program's code. Expanding the test suite for higher branch coverage may improve traditional testing outcomes but does not directly influence the thoroughness or accuracy of model checking outcomes.

+1 stayed within 4 sentences or less

-(1/2) went over 4 sentences



```
        | <-- (X) { result = ______ }       |
|
        |                                   |
|
        |                                   |
|
        |                                   |     (Y) { result = ______ }
--> |
        |                                   |
|
        |                                   |
|
        |                                   V
|
        .---------------------> (F) [ result.upper() ]
<---------------------.
                                            |
                                            |
                  (Z) { result = ______ } -> |
                                            |
                                            V
```

The control flow graph above corresponds to a short Python code snippet. We want to use a dataflow analysis to detect possible NoneType errors (i.e., any uses of, or operations on, a variable that has value None). In your response, answer the following questions:

(f) (2 points)

What direction of data analysis, forward or backward, should we employ to determine if there is a possibility that a NoneType error could occur (i.e., can the assignment at point A reach point F)? Justify your answer in 4 sentences or less.

ANSWER:

+1 Forward Analysis

+1 Reasoning - forward analysis can provide information about some code to "future" code -- along the normal path of execution. This the case for this problem where we are trying to figure out if some assignment reaches some other future point in the code.

(g) (3 points)

Consider the constant propagation dataflow analysis applied to the control flow graph above. Use the transfer functions discussed in class to determine what final dataflow value would be associated with “result” at each of (X), (Y), and (Z). Possible dataflow values in this context are: top, bottom, constant. Write your answer as three words separated by commas. The first word corresponds to your answer for (X), the second for your answer for (Y), and the third for your answer for (Z).

ANSWER:

+1 per each that is correct

Answer: [constant, constant, top]

You are eliciting requirements for a company making an app to handle applications to be a teaching assistant for EECS 481. Students who have taken the class before can submit an application to be considered. The customer tells you the following four statements: “(1) Applications must be submitted by December 21st 5PM Eastern Time to be considered. (2) If forms have any misspelling, it is automatically removed from the pool. (3) Applications must contain a 5000-word essay. (4) When rendered in 12pt font, essays must fit on a single 8.5x11 inch page with 1 inch margins.”

(h) (2 points)

Does the first customer statement have a terminology, designation, structure inconsistency, or no inconsistency with the second statement? Use only statements (1) and (2) when answering this. Support your claim in 3 sentences or less.

ANSWER:

+1 identifes it as a terminology issue

+1 Reasoning - the conflict occures between statements [1] and [2]. Specifically there is the use of the word "application" and "form". The customer is using them to refer to the same thing which is the "TA application" but never says this.

(i) (2 points)

Do any of the customer's statements have a strong conflict, weak conflict, or neither? Support your answer in 4 sentences or less.

ANSWER:

+1 Identifies there is a strong conflict

+1 Reasonning - the conflict occurs between [3] and [4]. It is not feasible to make an essay that is 5000 words long but also be rendered in the format the customer wants. This is a strong conflict becuase this conflict will presist for as long as the customer is accepting applications. This is not a temporary issue.

(j) (2 points)

What strategy would you use to resolve these sorts of conflicts, and why? Support your answer in 3 sentences or less.

ANSWER:

+2 Answers may very - Justifiable strategy to solve the conflict. Example: In this case, asking the customer to clarify the importance of having a certain word limit and page rendering. It is important to understand why the customer would like these things to be the same and if its a quality concern.

Question 4. Delta Debugging (10 points)

```
1
FIRST=0
2
SECOND=0
3
for i in $* ; do
4
  if [ $i -eq 5 ]; then FIRST=1 ; fi
5
  if [ $i -eq 8 ]; then SECOND=1 ; fi
6
```

```
done
7

8
if [ $FIRST -eq 1 ]; then
9
 if [ $SECOND -eq 1 ]; then
10
   exit 1 # yes, this set is interesting
11
 fi
12
fi
13

14
exit 0 # this set is not interesting
15
```

(a) (2 points)

Above is a bash script is-interesting.sh which describes one particular definition of “interesting” for Delta Debugging. Given the above bash script, how many tests (probes, considered subsets, calls to is-interesting.sh) does the Delta Debugging algorithm perform to identify the minimal subset when applied to input_list = [4, 5, 6, 8, 9, 10] ? Assume that in the case of an odd sized set, the split will result in the first half being smaller. Your answer should be just a number (no spacing or other characters).

ANSWER: 6

(b) (2 points)

With the above interesting definition, give an input_list such that Delta Debugging will encounter interference exactly ONE time. Remember that the input list must be interesting itself. Format your answer in the form of a Python list. (e.g., [1,2,3]). If it’s not possible, type “NOT POSSIBLE”.

ANSWER:

+2 Giving a valid list. As long as the list of integers contains "5" and "8", it is correct. Example: [4,5,8]

(c) (2 points)

With the above interesting definition, give an input_list such that Delta Debugging will encounter interference TWO OR MORE times. Remember that the input list must be interesting itself. Format your answer in the form of a Python list. (e.g., [1,2,3]). If it's not possible, type "NOT POSSIBLE".

ANSWER:

+ 2 NOT POSSIBLE

(d) (4 points)

Delta Debugging can identify a minimal set of conditions that cause a failure in a program. In the context of *Andreas Zeller's Automated Debugging: Are We Close?*, explain how Delta Debugging contrasts with traditional debugging methods and why it is considered more systematic and efficient in some cases. Support your answer with a relevant quote from the text using no more than 6 sentences.

ANSWER:

+1 Valid quote from source that complements reasoning

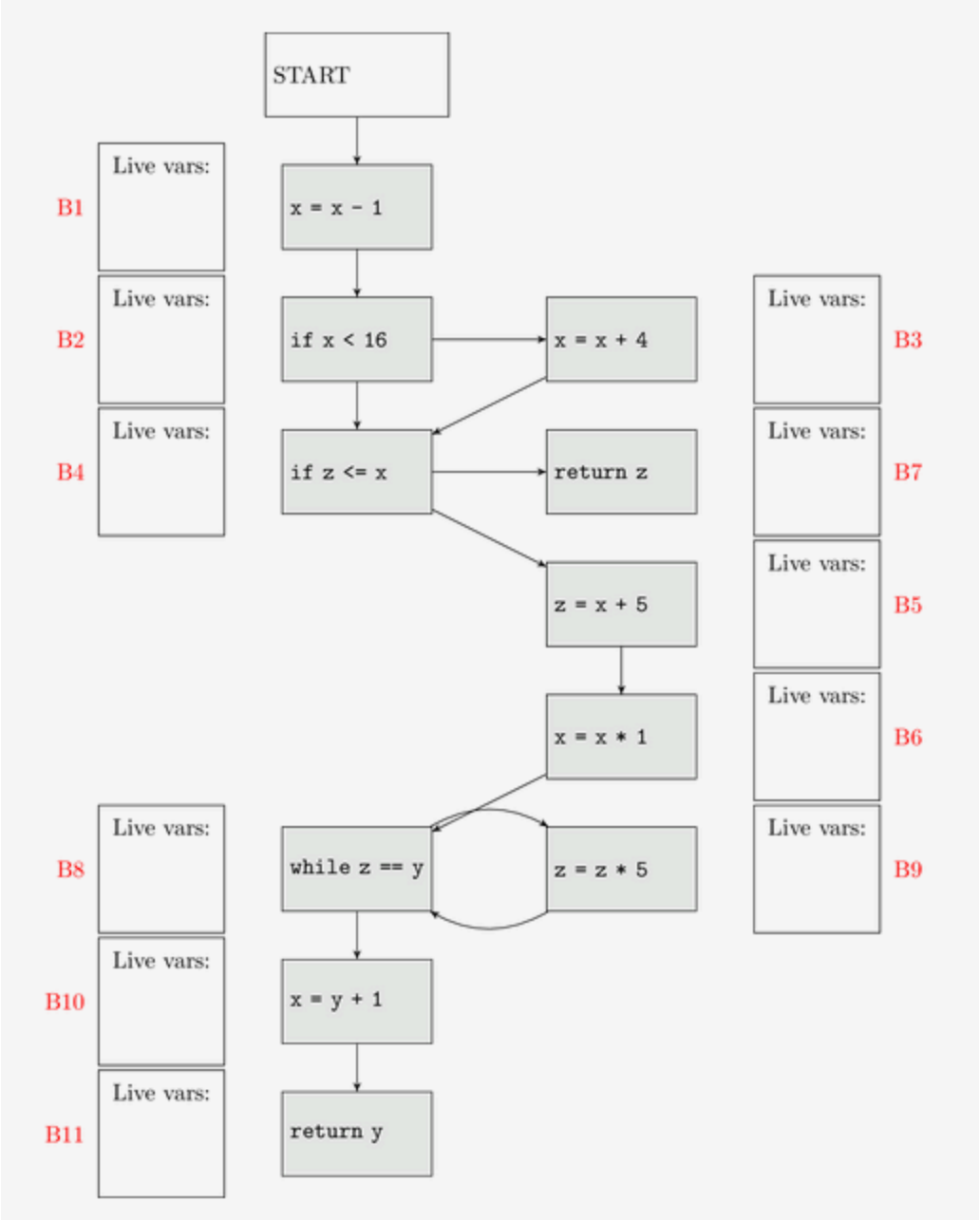
+2 Valid reasoning - look below for sample answer

+1 Stays within 6 sentences

Answer: Delta Debugging contrasts with traditional debugging by automating the process of narrowing down the minimal set of conditions that cause a failure, making it more efficient in many cases. In traditional debugging, programmers manually have to analyze code and test scenarios, which can be time consuming and error prone. As mentioned in the reading, "Delta Debugging always produces a set of relevant failure-inducing circumstances, which offer significant insights into the nature and cause of the failure." This systemic approach can reduce trial and error and provide a clearer understanding of the failure's root cause.

Question 5: Dataflow Analysis (16.5 points)

Consider a *live variable dataflow analysis* for three variables, `x`, `y`, and `z` used in the control-flow graph below. We associate with each variable a separate analysis fact: either the variable is (1) possibly read on a later path before it is overwritten (live), or (2) it is not (dead). We track the set of live variables at each point: for example, if `x` and `y` are alive but `z` is not, we write `{x, y}`. The special statement `return` reads, but does not write its argument. In addition, `if` and `while` read, but do not write all of the variables in their predicates. (You must determine if this is a forward or backward analysis.)



(1.5 points each) For each basic block B1 through B11, write down the list of variables that are live *right before* the start of the corresponding block in the control flow graph above. Please list only the variable names in lowercase without commas or other spacing (e.g., use either ab or ba to indicate that a and b are alive before that block).

B1

ANSWER: {'x', 'y', 'z'}

B2

ANSWER: {'x', 'y', 'z'}

B3
ANSWER: {'x', 'y', 'z'}

B4
ANSWER: {'x', 'z', 'y'}

B5
ANSWER: {'x', 'y'}

B6
ANSWER: {'x', 'y', 'z'}

B7
ANSWER: {'z'}

B8
ANSWER: {'y', 'z'}

B9
ANSWER: {'y', 'z'}

B10
ANSWER: {'y'}

B11
ANSWER: {'y'}

Question 6. Automatic Program Repair (16 points)

Automatic Repair Tools (like Repairnator or GenPRog or SapFix) and recent Large-Language Models (like ChatGPT or Codex) are algorithms that have been used to produce patches for buggy programs. We consider 4 concepts associated with these tools: fault localization, abstract syntax tree edits, machine learning, and maintenance costs. For each concept, identify the general approach (APR or LLMs) that uses, relates to, or improves that concept the MOST and describe how it does so in 4 sentences or less. In addition, you must support each claim with a separate quote from the *Automatic Program Repair* slides or *AI for SE* slides or *Monperrus et al.'s Repairnator patches programs automatically* reading.

(a) (2 points)

Fault Localization

ANSWER:

+(1/2) Identifies APR

+1 Reasoning- Automated Program Repair (APR) tools like GenProg and Repairnator rely on explicit fault localization techniques to identify buggy program components. They often use techniques like spectrum-based fault localization. LLMs do not use fault localization techniques and often attempt repairs based on broader contextual understanding of the code.

+(1/2) Valid Quotes from one of the three slides + complements their reasoning

---

(b) (2 points)

Abstract Syntax Tree Edits

ANSWER:

+(1/2) Identifies APR

+1 Reasoning- tools like SapFix and Genprog frequently generate and modify patches at the level of Abstract Syntax Trees. These tools leverage the formal structure of ASTs to ensure that patches adhere to the programming language's syntax rules. This approach allows APR to produce highly targeted and valid edits, often guided by predefined templates or learned repair patterns.

+(1/2) Valid Quotes from one of the three slides + complements their reasoning

---

(c) (2 points)

Machine Learning

ANSWER:

+(1/2) Identifies LLMs

+1 Reasoning- Large Language Models (LLMs), like ChatGPT and Codex, use machine learning techniques to address program repair tasks. By being trained on extensive datasets of code, they can create patches that may align with coding patterns and common bug resolution strategies. On the other hand, Automated Program Repair (APR) tools rely on fixed rules or heuristic-based searches (Fault localization, Fitness Function, etc.).

+(1/2) Valid Quotes from one of the three slides + complements their reasoning

---

(d) (2 points)

Maintence Costs

ANSWER:

+(1/2) Identifies APR - The context here is Maintence Costs associated with patch generation for fixing buggy programs. APR aims to directly improve this than tools such as LLMs

+1 Reasoning- Both APR tools and LLMs aim to reduce maintenance costs, but APR tools address maintenance costs associated with patching buggy programs directly by automating patch generation. APR approaches like Repairnator are made to try to reduce maintenance costs by automating bug fixes via techniques such as fault localization, AST edits, and testing - all of which are lacking in most LLMs used for patch generation and thus are not as effiecent or "geared" towards fixing of faulty programs.

+(1/2) Valid Quotes from one of the three slides + complements their reasoning

---

(e) (4 points)

Are tools that use the notion of “Generate and Validate” (like Repairnator) static or dynamic analysis? Are tools like LLMs used to repair buggy programs static or dynamic analysis? Explain in 6 sentences or less. In addition, you must include a quote from the *Automatic Program Repair* slides or *AI for SE* slides or *Monperrus et al.'s Repairnator patches programs automatically* reading to support your claim.

ANSWER:

+1 "Generate and Validate" Notion is dynamic analysis

+1 LLMs as program repair is static analysis

+1 Reasoning (+ 1/2 if only half the reasoning is present)- APR use the notion of "Generate and Validate" which involves the process of of generating patches and validating them by running them on a test suite. This is how plausible patches are determined and thus is considered dynamic analysis. LLMs, however, do not run the program or test their proposed patch on a test suite. Thus, these are static analysis.

+1 Valid quote from either valid resource + complements their reasoning

-(1/2) Over 6 sentences

(f) (4 points)

“Hallucinations” are programming outputs that, although seemingly plausible, deviate from users’ intent, factual knowledge, or contexts. Both APR and LLMs tool can generate candidate patches that might be hallucinations. Pick a broad category of tool (APR or LLM) and give a concrete example of the sort of hallucination it might create. Explain why this might happen in 6 sentences or less. In addition, you must include a quote from the *Automatic Program Repair* lecture slides or *AI for SE* slides or *Monperrus et al.'s Repairnator patches programs automatically* reading to support your claim.

ANSWER:

+2 Valid concrete/specific example of a “Hallucinations” within programming (if, instead, the example is broad -> +1) - APR: Tests- “compare yours.txt to trusted.txt”, GenProg's fix- “delete trusted.txt, output nothing”. OR LLMS: Tests- “compare yours.txt to trusted.txt”, ChatGPT's fix -"may have a syntax error but it presents it with confidence that its the right fix".

+1 Reasoning (+ 1/2 if only half the reasoning is present) - APR tools can make a patch that appears to work (passes current test cases), but may not implement the desired behavior. It could be that more test cases are need to reach that. LLMs, on the other hand, can present patches that fail the test cases but are presented as valid working code.

+1 Valid quote from either valid resource + complements their reasoning

-(1/2) Over 6 sentences

Extra Credit

(1) What was your favorite topic or activity during the course? (1 point)

---

(2) 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)? (1 point)

---

(3) Identify a different single optional reading (anything with the phrase *optional*) 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. (1 points)

---

(4) List one thing you learned and liked from ANY of the 4 guest speakers. Convince us that you paid careful attention during that lecture. For full credit, your answer must be different from your responses in the exam AND you must identify the guest speaker by name. (1 points)

---

(5) Identify ANOTHER 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. (1 points)

---

(6) Identify a course non-professor staff member (e.g., Priscila, Hanchi, Youcef, Leena, Livia, Christina, Rohit) by name and either describe one instance in which you had a positive interaction with that person or describe a potential area for improvement for that person as an instructor. (We take these comments seriously and use this information to determine who we ask back next year and to put people up for awards and recognition.) (1 points)

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.
Note that your submission will be marked as late. You can still submit, and we will retain all submissions you make, but unless you have a documented extenuating circumstance, we will not consider this submission.

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.