LATE

minutes remaining

Hide Time

Manual Save

Navigation

Question 1. Word Bank Matching (1 point each, 11 points total)

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. — Bottom-up Comprehension	B. — Bug Bounties	C. — Delegation	D. — Delta Debugging
E. — Functional Requirement	F. — GNS Theory	G. — Medical Imaging	H. — Mutation Testing
I. — Observer Design Pattern	J. — Profiling	K. — Program Synthesis	L. — Quality Requirement
M. — Quantum Computing	N. — Requirements Elicitation	O. — Risk Assessment	P. — Singleton Design Pattern
Q. — Stakeholders	R. — Top-down Comprehension	S. — Traceability	T. — Validation
U. — Verification			

Q1.1: The EECS 481 instructors decide to improve the online course webpage next semester. They identify many people who might care about the quality of the webpage and thus might be worth consulting as part of the work: junior and senior students who might take the course, instructional aides who might serve on the course staff, external tutors who might help students with the material, and graduate students who might take the course as a refresher.

Q1.2: Raffy wants to ensure that the requirements for a software project are correct, complete, and consistent.

Q1.3: When designing an autopilot system, a group of experts went through *this* procedure to identify potential failures in the system, the potential impacts of such failures, as well as response strategies for these failures in case they occur.

Q1.4: Daniel is implementing a game. Daniel wants to add such a feature: whenever the player is hit, the game would trigger some kind of animation and change the health bar in some way. The implementation will have classes for players, animations, and the health bar. Daniel can use *this* technique to implement the aforementioned feature with better extensibility (e.g., perhaps later a feature will be added to play a certain kind of sound as well when the player is hit).

Q1.5: Henry has some test cases generated by a fuzz testing tool. Henry wants to use *this* technique to evaluate the quality of these generated test cases automatically.

Q1.6: Paulina is implementing a logging class. It logs information about the program execution and can be called anywhere in the program. Paulina wants to use *this* approach to design the logging class in such a way that only one single instance of the class can be created in the program.

Q1.7: When reading the code, Jing first looks for semantic cues — for example, by reading method names, variable names, and comments — to guide their understanding of the source code.

Q1.8: Suppose that a company Rooble received a bug report together with some test cases that can reproduce the reported bug in their latest software release. However, the bug was not triggered in an earlier version a month ago. Rooble plans to use *this* algorithm to systematically narrow down to a small set of changes made in the past one month that can reproduce the bug, to better help them fix the bug.

Q1.9: BuzzyFuzzy is designing an online ordering system for restaurants. They have desire that the system must be fully operational 99.999% of the time and that the system must be able to handle all requests within 2 seconds.

Q1.10: Kris wants to scrape a large amount of data from a list of websites but unfortunately does not have any programming background. They can manually perform all of the scraping work, but that would be very tedious and time-consuming. Kris decided to use *this* technique which can automatically generate the desired program from high-level instructions on how to perform the task.

Q1.11: Bill is hired to write fluid simulation software for a group of scientists. Bill talks to some of the scientists to gather their opinions on what kind of functionalities this software should have, what UI and interactions they need, and how they will eventually use this software.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

Question 2. Delta Debugging (20 points)

For a particular multithreaded codebase, every thread either writes to variable x or reads x (but not both). That is, if a thread reads x, then it doesn't write to x. Likewise, if a thread writes to x, then we know it does not read x.

Given a finite set of threads from the codebase, there are different ways to schedule them. For this problem, we call a thread schedule *harmful* if, according to the schedule, x is read before any thread writes to x.

We consider a partitioned thread schedule made up of two sets of threads: *first* and *second*. The system will execute the threads in *first* in some order, and only once those are complete will the system execute the threads in *second* in some order. As an example, if *first* contains two threads, `i` and `j`, and *second* contains three threads, `a`, `b`, and `c`, then `[j, i, b, a, c]` is a possible thread schedule but `[b, i, j, a, c]` is not.

We consider a particular `interesting` function that takes as input the *second* set of threads. (The *first* set can be calculated by taking all of the threads in the system and removing those in *second*). The interesting function will **non-deterministically** pick a schedule that first executes all threads **not** in its input set (i. e., not in the *second* set) and then executes all threads in the input set (i. e., in the *second* set.) Then, `interesting` returns true if and only if the chosen thread schedule is not harmful.

In other words, the overall schedule is to run threads from the input *second* set **after** all threads outside the input *second* set have been executed. However, within each set, the threads can be scheduled in order.

Now, given this interesting function that takes as input the *second* set of threads and returns true/false, you are trying to find a smallest input *second* set (i. e., with the minimum number of threads) that is interesting (i. e., not harmful). The total set of threads is known and constant (e. g., `a, b, c, i, j` in the example above).

(a i) (3 points) When the input *second* set contains at least all threads that read x, the return value of the interesting function is true:

- ◯ Always
- ◯ Sometimes
- ◯ Never

(a ii) (6 points) If the delta debugging algorithm is run using the interesting function described above, do the following assumptions hold? Justify your answer. Use 1 sentence per assumption (your answer should be 3 sentences in total). Please write each sentence in a new line.

- (2 points) monotonicity (**Yes/No**)
- (2 points) unambiguity (**Yes/No**)
- (2 points) consistency (**Yes/No**)

Your answer here.

(a iii) (5 points) Now, the interesting function is **changed**: it runs *all possible* thread schedules (with threads in the input *second* set still always scheduled last) and it returns true if and only if no thread schedule is harmful. In this sub-question, we assume there exists at least one thread that writes to x and there is always at least one thread **not** in the input *second* set (in other words, the *first* set is always non-empty). Does delta debugging succeed (**Yes/No**) in this case? Justify your answer. Use no more than 3 sentences in your answer.

Your answer here.

(b) (3 points) Leaving aside this particular thread-scheduling context and considering general software engineering, would delta debugging still be useful if its time complexity were O(n log n) instead of O(log n) (**Yes/No**)? Justify your answer. Use no more than 3 sentences in your answer.

Your answer here.

(c) (3 points) In general software engineering, would delta debugging be useful in fuzz testing (**Yes/No**)? If so, how would you use delta debugging? Otherwise, why not? Use no more than 3 sentences in your answer.

Your answer here.

Question 3. Short Answer (3 points each, 15 points)

(a) (3 points)

C is considered to be a low-level language while Python is considered to be a high-level language.

What are two key reasons, with respect to productivity, that one would advocate for using a higher-level language? You may consider any aspects of software development (e.g., productivity, quality assurance, etc.). Limit your answer to no more than 4 sentences.

Your answer here.

(b) (3 points)

Your team is implementing a function that searches for an object from a large database.

This function takes as input two parameters: the target object and the database. Currently the two parameters of this function are named "x" (for the target object) and "y" (for the database). However, one of your colleagues suggests naming the two parameters as "target" and "database".

Do you agree or disagree with this idea and why? Please reference at least two pieces of evidence from the course or elsewhere when justifying your answer. Limit your answer to at most 4 sentences.

Your answer here.

(c) (3 points)

Automated Program Repair tools often generate multiple candidate mutants — from an original, buggy program — in hope that at least one of the candidate mutants will fix the bug in the original program.

For a candidate to be a valid plausible repair, it needs to pass the entire test suite. The entire test suite may consist of thousands of test cases and thus may take a long time to run.

How can dataflow analysis for dead code be used to make the testing process more efficient, and how is this related to the notion of static analyses being conservative? Please limit your answer to no more than 4 sentences.

Your answer here.

(d) (3 points)

Excel's FlashFill feature works well at converting a list of full names to a list of initials.

However, it does poorly when used to convert between a month's abbreviation to its full name (e.g., MAR to March). Explain why it fails in this case, using concepts covered in the lectures. Limit your answer to no more than 4 sentences.

Your answer here.

(e) (3 points) What is one tool that we can use to debug a multithreading program that we covered in lectures (you may reference any course material, including non-guest lecture material)? What is one effective technique Samyukta Jadhwani mentioned in her guest lecture?

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

Question 4. Fault Localization and Profiling. (20 points)

EECS 481 uses an exam server to host online exams where every student gets a different set of questions. To generate the content for a given question topic (e.g., the "Short Answers" topic), the TA will write a script that takes as input the student's information (e.g., their unique name) and returns specialized or randomized question content for that particular student.

Suppose you're one of the TAs who is responsible for writing the script for the "Short Answers" question. Specifically, each "Short Answers" question must be generated to contain exactly *two* sub-questions. You have prepared six possible sub-questions in total: DynamicAnalysisQuestion, MutationTestingQuestion, RaceConditionQuestion, MockingQuestion, StaticAnalysisQuestion and PairProgrammingQuestion. In your script, each of these six sub-0questions has a corresponding function that generates the content for that sub-question. For example, the "DynamicAnalysisQuestion" function generates a sub-question about dynamic analysis. You have also written a script that picks two sub-questions based on the student's unique name. The script is given below.

```
def generate_short_answers(string uniqname):
      print(uniqname)
      // pick a sub-question based on uniqname
      if (uniqname[0] == 'h'):
          q1 = DynamicAnalysisQuestion(uniqname)
      elif (uniqname[0] == 'x'):
          q1 = MutationTestingQuestion(uniqname)
      else:
          q1 = RaceConditionQuestion(uniqname)

      if (uniqname == "weimerw" || uniqname == "xwangsd"):
          q2 = MockingQuestion(uniqname)
      elif (len(uniqname) == 7):
          q2 = StaticAnalysisQuestion(uniqname)
      else:
          q2 = PairProgrammingQuestion(uniqname)
  return q1, q2


```

(a) (5 points)

After writing up the script, you want to test its correctness. In particular, you run it to generate the "Short Answers" questions for all students enrolled in 481 and see if the generated content can be correctly rendered by the exam server. Unfortunately, some of them failed: for some students, the content generated by your script cannot be correctly rendered. It may be an issue with the generate_short_answers function. Or, there may be a bug in one of the functions called by `generate_short_answers`.

Suppose you observed the following results. You want to use the concept of "Suspiciousness Ranking" from the Fault Localization Lecture to locate the lines that are more likely to contain the bug. In particular, we will consider only the lines given to you in the `generate_short_answers` function above.

Fill in the third column of the table with the lines visited by each of the inputs. To make it easier for us to auto-grade this question, please write your answer as **a list of line numbers in ascending order, separated only by a comma**. For example, to indicate only lines 1 and 2 were visited, write "1,2" (without quotes or spaces). Please do not include line numbers for code comments. We always count the first line of the program (i.e., line 1, the "def" line in the code snippet above) as visited.

Input	Correct Question Generated?	Lines visited (You fill this in!)
["henrybe"]	N	Your answer here.
["weimerw"]	Y	Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

[“xwangsd”]	N	Your answer here.
[“happy_student”]	Y	Your answer here.
[“DeDebug”]	N	Your answer here.

(b) (7 points)

Calculate the suspiciousness score for the lines in the table below. This question is going to be auto-graded, so please format your answer to **exactly two decimal places**. Please do not show work and do not round. For example, if the suspiciousness score is zero, enter "0.00" (without quotes, without spaces), and if the suspiciousness score is two thirds, enter "0.66" (without quotes or spaces).

Line	Suspiciousness Score
4	Your answer here.
5	Your answer here.
6	Your answer here.
7	Your answer here.
11	Your answer here.
12	Your answer here.
14	Your answer here.

(c) (3 points)

Which line(s) do you think are causing the problem (i.e., not rendering correctly), and why? Consider only the lines in the table from 6b. Please limit your answer to 2 sentences.

Your answer here.

Now suppose you have identified the bug and also fixed it. As a result, the script is now working correctly. However, it is much too slow. It took almost 1 hour to generate the exam for 100 students and EECS 481 has 10,000 students this semester! (This semester is the far-flung future of Fall 2050.) The TAs discussed and decided to profile the exam generation program to see which functions are slow.

In particular, you ran the main function (i.e., the entry function of the entire exam generation program) using a statistical profiler, which gives you the following function call profile:

```
1 * main()
    2 * make_exam()
        2 * read_uniqname()
        3 * generate_questions()
        4 * generate_Q6()
    1 * print_exam()
        2 * read_uniqname()
        99 * insert_emojis()


Self times:
main() - 120s
make_exam() - 600s
read_uniqname() - 720s
generate_questions() - 320s
generate_Q6() - 180s
print_exam() - 100s
insert_emojis() - 50s

```

(d) (5 points)

Here, the "self time" is the amount of time that's spent on the function **itself** excluding time taken by its (transitive) callees. For example, if a function `Foo` calls two functions `Bar` and `Baz`, the self time for `Foo` includes only time executing instructions in `Foo`, excluding `Bar` and `Baz`. We also assume that self times remain constant across runs: that is, running the function multiple times gives you the same self time for each run.

Please list the names of each function **in descending order** of probability that a random probe of the profiler would interrupt the program directly in that function. Interruptions occur directly in the function. For example, the probability of interrupting during `make_exam` involves code directly inside of `make_exam`, rather than code inside function calls made by `make_exam`.

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"] (including the brackets and quotes, but without spaces). Your list should contain all 7 function names (that is, the list should be of length 7).

Your answer here.

Question 5. Design Patterns (18 points)

Consider the following C++ code snippet that is used to reverse a singly-linked list. This function, named `reverse_linked_list`, is only relevant for questions 5a and 5b. The remaining questions (5c, 5d, 5e, 5f) are not necessarily based on the following code snippet.

For the following code snippet, you may assume the following:

- The Node and Linked List data structures are implemented correctly
- The `reverse_linked_list` function may be called on a linked list that contains zero or more nodes
- All functions not defined but involved in the code are implemented correctly

```
struct Node {
    int data;
    struct Node* next;
    Node(int data)
    {
        this.data = data;
        this.next = NULL;
    }
};

struct LinkedList {
    Node* head;
    LinkedList() { this.head = NULL; }
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

```
14
15      void reverse_linked_list()
16      {
17          Node* current = head;
18          Node* prev = NULL,
19          Node* next = NULL;
20
21          while (current != NULL) {
22              // Loop invariant is true here
23              next = current->next;
24              current->next = prev;
25              prev = current;
26              current = next;
27          }
28          head = prev;
29      }
30  };
```

(a) (3 points)

List two invariants of the `reverse_linked_list` function. The invariant should be true inside the loop at the line indicated by the comment, which reads "Loop invariant is true here". Do not restate any of the three assumptions listed above. Do not list a predicate that is trivial or true for all programs (e.g., 1+1=2). **Format your answer as two predicates separated by a newline.** In other words, rather than answering "P, Q", we would like you to put P and Q each on their own lines.

Your answer here.

(b) (3 points) List two post-conditions of the `reverse_linked_list` function. The post-condition should be true as the function terminates (either after a return, an uncaught exception, or executing the final line). Do not restate any of the assumptions listed above. Do not list a predicate that is trivial or true for all programs (e.g., 1+1=2). **Again, Format your answer as two predicates separate by a newline.** In other words, rather than answering "P, Q", we would like you to put P and Q on their own lines.

Your answer here.

(c) (3 points) Consider the Template Method Pattern and the software engineering goal of Designing for Code Comprehension. Argue whether or not that pattern specifically supports that design goal. Devise a brief example setting to support your argument. Additionally, call out at least two properties of the pattern and two aspects of the goal. Use at most 5 sentences.

Your answer here.

(d) (3 points) List 2 structural design patterns mentioned in the lecture and explain each of them in your own words. For each design pattern, also provide a small example — you can describe an example in the context of the LinkedList code snippet above, but feel free to give other examples too. For each design pattern (including your explanation and the example), please limit your answer to no more than 4 sentences. In total, your answer should use at most 8 sentences (4 sentences per design pattern, 2 design patterns in total).

Your answer here.

(e) (3 points) Use an example to explain why an anti-pattern can have a negative impact on software maintenance. Describe how you might redesign the original code to improve maintainability. Use at most 4 sentences total.

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

(f) (3 points) Consider the following git command: `git commit -m "did task 2, found and fixed task 1 bugs, tested task 2" (723 lines added, 48 lines removed)`. Do you believe this commit message follows good standards of code design and maintainability? If so, describe two design principles that are reflected in the message. If not, explain how you would fix the message and why. Use at most 4 sentences total.

Your answer here.

Question 6a. Requirements Elicitation (Part 1) (12 points)

Suppose you are asked by your manager to make an in-house version of Example-Guided Program Synthesis. This version should have as much of the functionality described in the lecture, reading, and/or industrial deployments as possible. To achieve this, you must write out some requirements for the application.

(a i) (4 points) List 2 functional requirements of the application. Each requirement should be one sentence.

Your answer here.

(a ii) (4 points) To help verify these functional requirements, you decide to write tests. For each functional requirement, what is one case you should test for? Your answer should be 4 sentences or less.

Your answer here.

(a iii) (4 points) List 1 verifiable quality requirement of the application. Then list 1 non-verifiable quality requirement of the application. Each requirement should be one sentence.

Your answer here.

Question 6b. Requirements Elicitation (Part 2) (4 points)

(b i) (2 points) Consider the two requirements below. What type of Flaw is present with these two requirements, and how might you combine them into one better requirement? Use 3 sentences or less.
1. There are no vehicles allowed in the amusement park. 2. In case of a medical emergency, an ambulance is allowed in the amusement park.

Your answer here.

(b ii) (2 points) List two desirable qualities of requirements, and briefly (in 4 sentences or less) explain why they are important.

Your answer here.

Question 7. Extra Credit (1 point each)

LATE

minutes remaining

Hide Time

Manual Save

Navigation

(*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 Sarah D'Angelo of Google that was not listed on an introductory summary slide or otherwise convince us that you paid careful attention during that lecture.

Sarah D'Angelo 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 Samyukta Jadhwani of Microsoft that was not listed on an introductory summary slide or otherwise convince us that you paid careful attention during that lecture.

Samyukta guest lecture

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.

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.