minutes remaining

Hide Time

Manual Save

Navigation

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. — Agile Development	B. — Alpha Testing	C. — Beta Testing	D. — Competent Programmer's Hypothesis
E. — Dynamic Analysis	F. — Formal Code Inspection	G. — Fuzz Testing	H. — Integration Testing
I. — Milestone	J. — Mocking	K. — Oracle	L. — Pair Programming
M. — Pass Around Code Review	N. — Perverse Incentive	O. — Priority	P. — Race Condition
Q. — Regression Testing	R. — Risk	S. — Sampling Bias	T. — Spiral Development
U. — Static Analysis	V. — Streetlight Effect	W. — Triage	X. — Unit Testing
Y. — Waterfall Model			

Q1.1:

J

A

Arjav wants to test the payment system for GlazeBook's new shopping app. They do not want to use real credit card numbers for privacy reasons, so they create a fake credit card and use that in their unit tests.

Q1.2:

W

B

After beta testing, Boogle has hundreds of bugs to fix. In order to determine which ones to prioritize first, they review each bug and assign a severity level to it based on the amount of revenue the bug could lose the company.

Q1.3:

U

C

Before contributing to a repository, MunchyRoll has tools that automatically check the changed code, without running it, to make sure certain properties are not violated.

Q1.4:

T

D

In order to make very important, confidential election software, 481co.inc continuously goes through cycles of identifying objectives, developing, testing, and reviewing with the goal of mitigating the amount of risk every time.

Q1.5:

Y

E

Arian is working on developing a chatbot. They decide to approach the development process by gathering requirements, designing prototypes, implementing the solution, testing, then finally deploying the product. They fully complete each step before moving onto the next.

minutes remaining

Hide Time

Manual Save

Navigation

Q1.6:

M

F

Netfleecs enforces all code changes must be reviewed by two other developers before being merged to the codebase. The reviewers get an email where they can inspect the changes asynchronously.

Q1.7:

X

G

In EECS 280, students have to create a Matrix class to work with a 2D array of pixel values. In order to make sure that all functions have the intended functionality, the students must write test cases specifically testing each function.

Q1.8:

Q

H

While building a calculator app, Yunchi implements the functionality for division operations. During testing, they run into an error where the application crashes when the user tries to divide by 0. Yunchi add's a check for this and writes a test to ensure that this problem does not come up again.

Q1.9:

R

I

Bad coding practices, poor planning, and a lack of code maintenance can introduce this undesirable quality into a project.

Q1.10:

O

J

Azure DevOps is a tool that allows developers to track their team's work items. You can assign a value from 1-3 (1 being immediately and 3 being optional) based on how quickly the problem needs to be addressed.

Q1.11:

F

K

When making an important change, developers at EECSon Mobile forgo the normal code review process and instead meet in person to review the code line by line, discussing any potential bugs.

Q1.12:

S

L

American game developer Bank of Michigan aims to release a game to the entire world, however, they were lazy when selecting testers and randomly selected people from the city they are headquartered in. As a result, the testers were heavily skewed to a single ethnicity.

Q1.13:

K

M

To test their calculator app's parenthesis functionality, Max writes a test case to see if performing "5-(4+2)" returns -1. Here, -1 is a particular example of this concept.

Q1.14:

G

N

Appa is creating a custom login form, and tests the password field by attempting to enter 10,000 strings of random characters to see if the form behaves as expected, does not crash, and is not vulnerable to SQL injection.

Question 2. Test Coverage Analysis (20 points)

minutes remaining

Hide Time

Manual Save

Navigation

You are given the following pseudocode for a program that determines if a player will play well in the upcoming season.

In this question, we assume statement coverage applies to **only** statements marked STMT_#, and we consider the entire program when calculating statement coverage. In other words, the program starts at `willPerformWell()`, but even if some methods are not executed during the program execution for a given input, we still consider coverage with respect to all STMTs in the entire program. Finally, please remember that you may need to scroll down to view the full program code

```
1  bool isRivalCity(string city) {
2      if (city == 'Columbus' || city == 'East Lansing' || city == 'Seattle') {
3          STMT_1
4          return false;
5      }
6
7      STMT_2
8      return true;
9  }
10
11 bool isPrime(int num) {
12     if (num == 1 || (num % 2 == 0 && num > 2)) {
13         STMT_3
```

(2.1) (4 points)

Choose the input that will achieve 37.5% statement coverage. If none can achieve it, choose *IMPOSSIBLE*.

- (●) A) `willPerformWell(50, 30, "Jack", "Columbus")`
- ( ) B) `willPerformWell(49, 10, "Montgomery", "Ann Arbor")`
- ( ) C) `willPerformWell(37, 10, "Lockhart", "Boston")`
- ( ) D) `willPerformWell(10, 2, "Lovelace", "Chicago")`
- ( ) E) `willPerformWell(47, 5, "Erickson", "Boston")`
- ( ) F) IMPOSSIBLE

ANSWER: C) willPerformWell(37, 10, "Lockhart", "Boston") --> 3 Statements (For those who saw the typo: E was also accepted)

(2.2) (4 points)

What is the **lowest** statement coverage that can be achieved with one input? Please (1) first select the statement coverage and (2) then select an input that will achieve that coverage. (2 points each)

- ( ) A) 12.5%
- (●) B) 25%
- ( ) C) 50%
- ( ) D) 75%
- ( ) E) 100%
- ( ) F) None of the Above

ANSWER: B) 25% is the lowest possible statement coverage

- ( ) A) `willPerformWell(37, 10, "Lockhart", "Boston")`
- ( ) B) `willPerformWell(49, 10, "Montgomery", "Ann Arbor")`
- (●) C) `willPerformWell(47, 5, "Erickson", "Boston")`
- ( ) D) None of the Above

ANSWER: B) willPreformWell(49, 10, "Montgomery", "Ann Arbor") will result in the lowest statement coverage

---

(2.3) (2 points)

In this question, we **only** consider branches created by the 5 if-statements in the program above (meaning there are 10 branches in total).

What is the **minimum** number of test cases needed to get **exactly** 30% branch coverage, with no restriction on the inputs? If you believe the given branch coverage cannot be achieved by any inputs, please choose IMPOSSIBLE.

- ( ) A) 1 Test Cases

- B) 2 Test Cases
- C) 3 Test Cases
- (●) D) 4 Test Cases
- E) IMPOSSIBLE

ANSWER: E) IMPOSSIBLE to get 30% branch coverage

minutes remaining

Hide Time

(2.4) (2 points)

Manual Save

In this question, we **only** consider branches created by the 5 if-statements in the program above (meaning there are 10 branches in total).

Navigation

What is the **smallest** branch coverage that could be achieved with a single test input?

- A) 20%
- B) 30%
- C) 50%
- D) 60%
- (●) E) 70%
- F) None of the Above

ANSWER: A) 20% branch coverage is the least we can get with one input

---

(2.5) (3 points)

(1) How many paths does your answer from 2.4 cover? (2) What is the maximum number of paths a single input can cover? (1.5 points each)

- (●) A) 1 Path
- B) 2 Paths
- C) 3 Paths
- D) None of the Above

ANSWER: A) One input only covers one path

- A) 1 Path
- (●) B) 2 Paths
- C) 3 Paths
- D) None of the Above

ANSWER: A) One input covers a max of one path

(2.6) (5 points)

Based on lecture content, define branch coverage and how it differs from statement coverage.
Why is branch coverage considered to be a more comprehensive testing strategy than statement coverage?
Please include an example that demonstrates how branch coverage reveals issues that statement coverage might miss.
Use no more than 6 sentences in your entire answer.

2.6ANSWER

ANSWER:
+2 Branch coverage definition

+1 How it differs from statement coverage

+2 Valid and specific example (1 pt if not specific enough)

Statement coverage → Helps identify "dead" or inaccessible code, but It doesn't really matter how the line is executed. We could give it a bunch of random input, with no reasoning, to try to maximize the coverage. Least thorough out of the three.

Branch Coverage → All branches will lead to 100% statement coverage. Ensure all branches and their alternatives have been run.

minutes remaining

Hide Time

Manual Save

Navigation

Question 3. Short Answer (25 points)

(a) (5 points)

You are working on a web application that is about to be released for public use. During the final testing phase, your team is concerned about ensuring the application handles user inputs correctly.

Identify two quality assurance techniques from the *Quality Assurance and Testing* lecture that could address this concern. Explain how each technique helps ensure the application is ready for release. Limit your answer to at most 2 sentences for each technique (that is, at most 4 sentences). (5 points)

3.a

ANSWER:
Answers may vary:

- +1 points for each technique
- +1 point for each corresponding explanation
- + 1 pt if both techniques are from *Quality Assurance and Testing* lecture

Sample Answer:
Use unit testing to test user inputs work correctly. Unit testing ensures each component that processes the input works correctly.

Use fuzz testing to ensure that bad inputs get handled correctly. This helps ensure that we have considered all possible user inputs and are handling invalid user inputs correctly.

(b) (5 points)

You are the CEO of a large software company and are looking for ways to assess employee performance in order to determine bonuses and promotions. You decide to use lines of code as the only metric to track employee production.

What is a shortcoming of using this metric? Give an example scenario of how using only lines of code does not accurately reflect employee performance and how we can improve our performance criteria in such a scenario. Limit your entire answer to at most 4 sentences.

3.b

ANSWER:
Answers may vary:

- +2 for flaw
- +1 for example scenario
- +2 for improvement

Sample Answer:
Lines of code does not account for code quality, documentation, or time spent on design. Additionally, code could consistently be very low quality and not ever reach production
Bob consistently puts out subpar code that his teammates do not approve in his code reviews, but this earns him a very high score in terms of lines of code produced

Modify criteria to be lines of code that reach production level, with slight penalties for code reviews that are rejected (in order to incentivize spamming code reviews)

minutes remaining

Hide Time

Manual Save

Navigation

(c) (10 points)

You are an engineering manager who must decide whether to employ pair programming for a series of tasks---see questions (ci)-(cii). Your plan is to use the type of reasoning from EECS 481 (such as the cost-benefit analysis between pair programming and solo programming) to help make the decision.

For each task, calculate the pair programming cost and solo programming cost, then determine their ratio (`pair programming cost / solo programming cost`). For example, if the pair programming cost is 91 and the solo programming cost is 100, the ratio would be 0.91. Pair programming is cost-effective if this ratio is 0.90 or lower ($\leq$0.90).

Follow this methodology to analyze the tasks below. Based on your calculations, determine whether the manager should use "pair" or "individual" programming for each task. **Note:** please format your answer as follows (otherwise our auto-grader might misgrade your answer).

Solo Programming Cost
Pair Programming Cost
Choice ("pair" or "individual")

For example, if the solo programming cost is 1400 and the pair programming cost is 1000, your answer should be:

1400
1000
pair

Make sure to write your answer rounded to two decimal places (e.g., 1.23). The interpretation of the various features (e.g., "fewer total lines" or "fewer total defects") should be as discussed in class, where factors like collaboration, error reduction, and time efficiency are taken into account. **Note: pair programming does not improve the speed of fixing defects—it only reduces the number of defects.**

(ci) (5 points)

Consider a program with 47,000 LOC. Suppose the coding speed for one programmer is 94 LOC/hour. The corresponding defect rate is 12 defects/KLOC. Suppose the defect fix time is 21 hours/defect. Pair programming results in 13% fewer total defect, but slows down the overall coding speed by 11%.

3.c1

ANSWER:
12344
10859.28
pair

(cii) (5 points)

Consider a program with 60,000 LOC. Suppose the coding speed for one programmer is 25 LOC/hour. The corresponding defect rate is 10 defects/KLOC. Suppose the defect fix time is 15 hours/defect. Pair programming results in 6% fewer total defect, but slows down the overall coding speed by 30%.

3.c2

ANSWER:
11400
11580
individual

minutes remaining

Hide Time

Manual Save

Navigation

(d) Answer questions (di)-(diii) using the following scenario:

Suppose you are asked the following question during a technical interview (the particular programming language isn't relevant for this problem): Given a string of size n containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring. For example, "(())" is valid, but "(()" is not.

(di) (2 points)

Write two questions from slide 45 of the *Pair Programming and Skill Interviews* lecture that you could ask the interviewer before beginning your implementation. If none are relevant, explain why in no more than three sentences.

3.d1

ANSWER:
+1 for saying no valid question
+1 for valid reasoning
Example Answer: The questions on the slide focus on asking about the array and the array elements, which are integers. In this question we are given a string with only 2 characters, so the questions don't provide us any useful information about the string.

(dii) (2 points)

Given the following answer to the question above. Identify 2 lines in the code that could benefit from comments. Provide the line numbers and what the comments should be.

```
1  int longestValidParentheses(string s) {
2          int ans = 0;
3          stack<int>st;
4          st.push(-1);
5
6          for (int i = 0; i < s.size(); ++i) {
7              if(s[i] == '(') {
8                  st.push(i);
9              }
10             else {
11                 st.pop();
12                 if(st.empty()) {
13                     st.push(i);
```

3.d2

ANSWER:
Answers may vary:
+1 points for each comment

Sample answer:
line 6 # Store index of '('
line 15 Calculate the length of the current valid substring using the top index of the stack

minutes remaining

Hide Time

(diii) (1 points)

After writing the code, the interviewer asks you to provide the time complexity of your solution in Big O notation. Write in terms of n, the size of the input string.

- (●) A) O(n)
- ( ) B) O(1)
- ( ) C) O(n^2)
- ( ) D) O(log n)

ANSWER: O(n)

Manual Save

Navigation

Question 4. Mutation Testing (15 points)

The following program takes an integer as input and returns the number of unique prime factors of the input.
Each mutant represents a **single-character** modification to the code that may or may not impact the correctness of the function. You can assume that the unmodified implementation is correct and returns the correct number of prime factors for all positive integers. Only one mutant is applied at a time.
Note that this program invokes a function called `int(a)`, which rounds down the input `a` to the nearest integer.
Also, you may need to scroll down to view the full program code

```
1  def exactPrimeFactorCount(num) :
2      n = num
3      count = 0
4      if (n % 2 == 0) :                # Mutant 1 -> n % 2 == 1
5          count = count + 1
6          while (n % 2 == 0):          # Mutant 2 -> while (n % 2 != 0)
7              n = int(n / 2)
8
9      # n must be odd at this point
10     i = 3
11
12     while (i <= int(math.sqrt(n))):  # Mutant 3 -> i < int(math.sqrt(n))
13         if (n % i == 0) :
```

(4a) (8 points)

For each of the inputs, `num`, to the `exactPrimeFactorCount` function above, determine whether each mutant is killed (True) or not (False). The oracle represents the result of the unmodified function.
Reminder: a mutant is killed by a test case if that test case produces different results on the mutated and unmutated code.

	Oracle	Mutant 3	Mutant 2
n = 3	1	(4a_1) (1 points) (●) True ( ) False ANSWER: False	(4a_2) (1 points) (●) True ( ) False ANSWER: False

minutes remaining

Hide Time

Manual Save

Navigation

	Oracle	Mutant 3	Mutant 2
n = 9	1	(4a_3) (1 points) ○ True ◉ False ANSWER: False	(4a_4) (1 points) ○ True ◉ False ANSWER: False
n = 6	2	(4a_5) (1 points) ◉ True ○ False ANSWER: False	(4a_6) (1 points) ○ True ◉ False ANSWER: False
n = 15	2	(4a_7) (1 points) ◉ True ○ False ANSWER: True	(4a_8) (1 points) ○ True ◉ False ANSWER: False

(4b_1) (2 points)

What is the mutation adequacy score for inputs n=3, n=9, and n=6 when Mutant 3 is applied? **Note:** please express your answer in **simplified fraction with no spaces**: for example, `1/2`. If your answer is 1 or 0, just use 1 or 0.

4b1

ANSWER: 0

(4b_2) (2 points)

What is the mutation adequacy score for inputs n=9, n=6, and n=15 when Mutant 2 is applied? **Note:** please express your answer in **simplified fraction with no spaces**: for example, `1/2`. If your answer is 1 or 0, just use 1 or 0.

4b2

ANSWER: 0

(4c) (3 points)

The following invariants are generated by a tool based on a set of test cases for `exactPrimeFactorCount`.
Assume no mutants are applied and that the input `n` is always valid (i.e., positive integer). The invariants are evaluated at the end of the function, and are considered valid if they always hold. For each invariant, select True to indicate that the invariant is valid and False otherwise.

(4c_1) (1 points) `n > 2`

◉ True
○ False

ANSWER: False

(4c_2) (1 points) `i > sqrt(n)`

- (•) True
- ( ) False

ANSWER: True

minutes remaining

Hide Time

Manual Save

Navigation

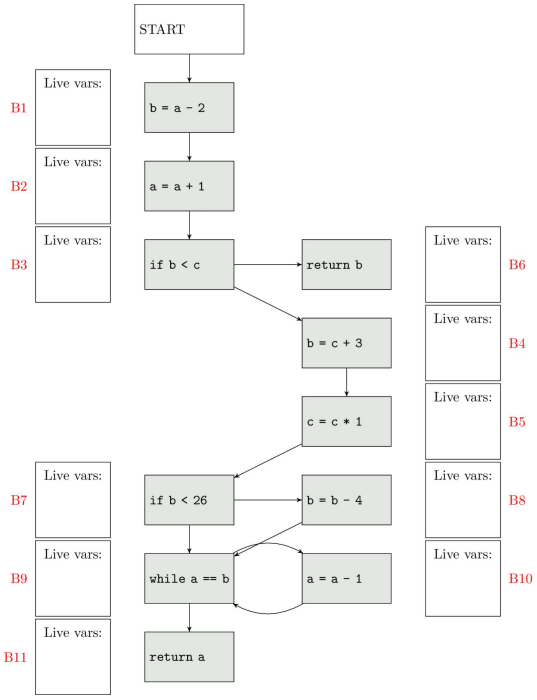
(4c_3) (1 points) `count >= 1`

- ( ) True
- (•) False

ANSWER: False

Question 5: Dataflow Analysis (11 points total)

Consider a *live variable dataflow analysis* for three variables, `a`, `b`, and `c` 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 `a` and `b` are alive but `c` is not, we write `ab`. 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 point 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 1

ANSWER: {'c', 'a'}

B2 2

ANSWER: {'c', 'a', 'b'}

B3 3

ANSWER: {'c', 'a', 'b'}

B4 4

ANSWER: {'c', 'a'}

B5 5

ANSWER: {'c', 'a', 'b'}

B6 6

ANSWER: {'b'}

B7 7

ANSWER: {'a', 'b'}

B8 8

ANSWER: {'a', 'b'}

B9 9

B10 10

B11 11

ANSWER: {'a', 'b'} ANSWER: {'a', 'b'} ANSWER: {'a'}

Question 6. Dynamic Analysis (15 points)

minutes remaining

Hide Time

Manual Save

Navigation

You have developed a dynamic analysis tool, called MemGuard, that aims to quickly identify potential memory issues, including double frees, missing deallocations, and use-after-free errors.

MemGuard works by tracking and logging memory allocations, deallocations, and memory accesses during program execution. To improve efficiency, MemGuard runs and analyzes multiple threads of the same program concurrently, assigning each thread a unique number (e.g., Thread 1, Thread 2, etc.).

However, the MemGuard instrumentation may be buggy, leading to issues in the log, such as:
(1) Missing events or details (e.g., missing deallocation logs).
(2) Spurious events (e.g., allocations or deallocations that never actually occurred).

MemGuard reports potential memory issues based on its analysis of the log file. These issues may lead to false positives or false negatives:
(a) False positive: MemGuard incorrectly reports a memory issue that **doesn't actually exist** in the subject program being analyzed.
(b) False negative: MemGuard fails to detect a memory issue that **is actually present** in the the subject program being analyzed.

Below are programs that MemGuard analyzed, along with the log files for each. For each program, you are asked to:
(i) Determine whether the program itself has any **actual memory issues** (e.g., double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.
(ii) Determine whether MemGuard would report any memory issues **based on the log file** (e.g, double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.
(iii) Determine if MemGuard produced any false positives or false negatives.

NOTE: You may need to scroll down to view the full program code or log file for some examples.

```
void p1() {
    int* ptr1 = new int;
    int* ptr2 = new int;

    *ptr1 = 42;
    *ptr2 = 84;

    if (*ptr1 == 84) {
         delete ptr1;
    }
    *ptr1 == 2;

    delete ptr1;
```

(ai) (1 points)

Determine whether the program has any of the following memory issues (e.g., double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.

- (●) A) No Error
- ( ) B) Missing Deallocation
- ( ) C) Use-After-Free
- ( ) D) Double Free

ANSWER: No error

(aii) (1 points)

Determine whether MemGuard would report any memory issues based on the log file (e.g, double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.

- ( ) A) No Error

- (●) B) Missing Deallocation
- ( ) C) Use-After-Free
- ( ) D) Double Free

ANSWER: Missing Deallocation

minutes remaining

Hide Time

(aiii) (1 points)

Determine if MemGuard produced any false positives or false negatives.

Manual Save

- ( ) A) Neither
- ( ) B) False Positive
- (●) C) False Negative
- ( ) D) Both

ANSWER: False Positive

Navigation


```
void p2() {
    int* ptr1 = new int;
    int* ptr2 = new int;

    *ptr1 = 42;
    *ptr2 = 84;

    if (*ptr1 == 42) {
        delete ptr1;
    }
    *ptr1 == 2;

    delete ptr2;
```

(bi) (1 points)

Determine whether the program has any of the following memory issues (e.g., double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.

- ( ) A) No Error
- ( ) B) Missing Deallocation
- ( ) C) Use-After-Free
- (●) D) Double Free

ANSWER: Use-After-Free

(bii) (1 points)

Determine whether MemGuard would report any memory issues based on the log file (e.g, double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.

- (●) A) No Error
- ( ) B) Missing Deallocation
- ( ) C) Use-After-Free
- ( ) D) Double Free

ANSWER: Use-After-Free

(biii) (1 points)

Determine if MemGuard produced any false positives or false negatives.

- ( ) A) Neither
- (●) B) False Positive
- ( ) C) False Negative

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

○ D) Both

ANSWER: Neither

```
void p5() {

    int* array1 = new int[5];

    int* array2 = new int[3];

    for (int i = 0; i < 3; ++i) {

        int* tempArray = new int[10];

        delete[] tempArray;
    }
```

(ci) (1 points)

Determine whether the program has any of the following memory issues (e.g., double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.

○ A) No Error
○ B) Missing Deallocation
◉ C) Use-After-Free
○ D) Double Free

ANSWER: No Error

(cii) (1 points)

Determine whether MemGuard would report any memory issues based on the log file (e.g, double free, missing deallocation, use-after-free). If multiple issues are present, report the first one that occurs.

○ A) No Error
○ B) Missing Deallocation
○ C) Use-After-Free
◉ D) Double Free

ANSWER: Missing Deallocation

(ciii) (1 points)

Determine if MemGuard produced any false positives or false negatives.

◉ A) Neither
○ B) False Positive
○ C) False Negative
○ D) Both

ANSWER: False Positive

(d) Answer questions (di)-(dii) using the following scenario:

You are debugging a multithreaded program experiencing race conditions that occasionally cause inconsistent results during file processing. To address the issue, you are considering using MemGuard, which you can assume functions correctly (i.e., it produces no false positives or false negatives).

---

minutes remaining

Hide Time

Manual Save

Navigation

(di) (3 points)

Evaluate the use of MemGuard to identify and resolve the race condition in your program. Explain how MemGuard could assist and provide an example scenario where MemGuard might not be effective. Limit your answer to at most 4 sentences.

d1

ANSWER:
Answers may vary

- +1.5 points for how MemGuard could assist
- +1.5 points for example scenario where MemGuard might not be effective

Sample Answer:
MemGuard is a dynamic analysis tool designed to detect memory access violations and race conditions by monitoring runtime memory operations. It can help identify memory management issues, which is a common cause of race conditions. However, MemGuard might not be effective if the race condition is unrelated to memory, such as one caused by improper thread coordination or logic errors. For instance, if a race condition arises from threads waiting incorrectly, MemGuard may not detect it.

(dii) (3 points)

What is one dynamic analysis tool you learned in class that could help identify the race condition in the program? Briefly explain how the tool works and how it would assist in finding the race condition. Limit your answer to at most 4 sentences.

d2

ANSWER: Answers may vary

- +1 point for naming a dynamic analysis tool talked about in class
- +1 point describing how the tool works
- +1 point describing how the tool would assist in finding the race condition

Sample Answer:
One dynamic analysis tool that could help is CHESS. CHESS systematically explores different thread interleavings to identify potential race conditions that occur due to non-deterministic thread scheduling. By running the program under multiple thread schedules, CHESS can expose situations where race conditions might manifest. CHESS can detect heisenbugs that depend on the exact timing and order of thread execution which could be the cause of the inconsistent file processing.

Extra Credit

(1) What is your favorite part of the class so far? (1 point)

ec1

(2) What is your least favorite part of the class so far? (1 point)

ec2

minutes remaining

Hide Time

Manual Save

Navigation

(3) In the context of HW2, how would the SAGE tool from the *"Automated Whitebox Fuzz Testing"* paper compare to EvoSuite on jsoup? Demonstrate that you have read the paper critically and tie it in to your experiences with HW2 (i.e., what did EvoSuite do badly at for 481 specifically?), going beyond a Generative AI summary. (2 points)

ec3

(4) If you read any *other* optional reading, identify it and demonstrate to us that you have read it critically, going beyond a Generative AI summary. (2 points)

ec4

(5) In at most 4 sentences, please share one of your key take-aways from the guest lecture by Natalia Sánchez Rocafort, and explain which part of her talk (e.g., by giving slide numbers, etc.) led to this take-away. (2 points)

ec5

(6) Did you use ChatGPT or any Generative AI tool on this exam? Do such tools help with this sort of exam? Should we allow ChatGPT on Exam #2? (Remember, free ChatGPT is allowed, so you're not cheating. This is to help us improve the course, not to get you in trouble.) (2 points)

ec6

(7) Attendance (either in person or remote) has been a requirement for EECS 481 since many years ago. In the past, we used physical notecards to collect student names for in-person participation. This term (Winter 2025) we are using the iclicker app. Do you find it is easy enough to use iclicker? Would you suggest keeping that for future offerings? (1 point)

ec7

Honor Pledge and Exam Submission

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

- [ ] I have neither given nor received unauthorized aid on this exam.
- [ ] *I am ready to submit my exam.*

Submit My Exam

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

The exam is graded out of 100 points.