LATE

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. — A/B Testing	B. — Agile development	C. — Alpha Testing	D. — Beta Testing
E. — Competent Programmer Hypothesis	F. — Constructive Cost Model	G. — Dynamic Analysis	H. — Formal Code Inspection
I. — Fuzz Testing	J. — Milestone	K. — Mocking	L. — Oracle
M. — Pair Programming	N. — Pass Around Code Review	O. — Priority	P. — Race Condition
Q. — Regression testing	R. — Risk	S. — Severity	T. — Spiral Development
U. — Streetlight Effect	V. — Triage	W. — Uncertainty	X. — Unit Testing
Y. — Waterfall Model			

Q1.1:

Miscord recently started working on a new project. Ben is in constant contact with the stakeholders, modifying the project as their needs change.

Q1.2:

EECSon Mobile has a policy that for every created function, there must be corresponding inputs, outputs and oracles to assess that function.

Q1.3:

Livia decides, on a whim, to watch the latest Parvel movie, Avengers: End of Germs. Upon arriving at the theater, they discover that someone else is already sitting in their assigned seat. It turns out both people bought tickets online at the last minute and were assigned the same seat.

Q1.4:

Aang is developing a social media app and designing the user interface. To verify that the interface displays information properly, they are simulating the backend data using a static JSON file.

Q1.5:

Veecsa is preparing to launch their latest application, Adobe Shotofop. Prior to making it available to the general public, they plan to have a group within the company evaluate it to uncover potential issues.

Q1.6:

Omkar has recently been brainstorming things that could go wrong in an effort to manage *this*.

Q1.7:

Boogle is finding maintenance to be a very expensive part of the development process. This is due to Boogle not accurately identifying the impact each bug has on the business.

Q1.8:

Antony is developing a program to determine whether a number is prime. They double-check corner cases, such as the inputs 0, 1 and 2, to be sure of what the answers should be.

Q1.9:

PiedPiper is developing a new software product and plans to repeat this process until completion: create a prototype, gather user feedback, mitigate risk, and make an improved prototype.

Q1.10:

FloorMart is a travel agency that arranges global tours. They are encountering problems where customers can book tickets for tours that are already sold out because multiple bookings are processed simultaneously. Harry proposes using *this* technique to identify the root cause of this issue automatically.

Q1.11:

The highly-anticipated game, AldenRing, is about to be launched. Six months before its release, Devforce distributes a pre-release version to a select group of users.

Q1.12:

Mesla Inc. recently found that developers were making commits directly to the main branch without input from others. To address this, they introduced a policy requiring at least one other developer familiar with the codebase to review and approve any changes before they are committed.

Q1.13:

MunchyRoll recently hired new developers who have expressed concerns about critical code being difficult to read and filled with bugs. To address this, Gwen, a senior developer, decides to conduct a series of structured meetings with the team to review and resolve the issues in that code.

Q1.14:

Yushu is working on a new game for GogoWeimer. GogoWeimer sets a goal to complete the physics module within the next two months.

Question 2. Code Coverage (25 points)

You are given the following code. (You can scroll down to see the all the code) In this question, we consider the entire program when calculating coverage.

```
1 void coverage(bool a, bool b){
2   if (a || b) {
3     std::cout << "1";
4   }
5   if (!b || !b || !b || !b) {
6     std::cout << "2";
7   }
8 }
9
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

(a) (4 points)

What is the minimum number of test cases required for 100% statement coverage? Enter a whole number.

Your answer here.

(b) (4 points)

What is the minimum number of test cases required for 100% branch coverage? Enter a whole number.

Your answer here.

(c) (4 points)

What is the minimum number of test cases required for 100% path coverage? Enter a whole number.

Your answer here.

Now consider the `silly_goose` function. Answer the following questions.

```
void silly_goose(bool a, bool b, bool c){
    if ((a || !b) || (c && !a)) {
        STMT_1;
    }
    if ((c && !b) || (a || b)) {
         STMT_2;
    }
    if ((!a && !b) && c){
        STMT_3;
    }
}

```

(d) (4 points)

How many of the STMT statements does the test case `(true, true, true)` cover? Write your answer as a whole number. (We are asking for the statement coverage without the denominator.)

Your answer here.

(e) (4 points)

How many of the branch directions does the suite `[(false, false, true), (false, true, false), (false, false, false)]` cover? Write your answer as a whole number. (We are asking for the branch coverage without the denominator.)

Your answer here.

(f) (5 points)

Would EECS481 Homework 1b have been easier or harder if students had only been asked to provide 1% *(acylic) path* coverage for full points (instead of the listed coverage requirements)? Justify your answer in 5 sentences or less. Your answer should reference details about the program in question.

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

Question 3. Short Answer (24 points)

This question (a-b) concerns bug triage. Consider the bug report problems from the reading *"What Makes a Good Bug Report?"*. For each bug below, list an **uncommon** problem (as reported by that reading) that could cause an organization to mistakenly assign the bug report the wrong priority, then give a potential example of such a scenario using a real, specific tool or assignment from EECS 481 as a basis.
For each bug, please limit your answer to at most 4 sentences for that bug.

(a) (3 points)

A bug that causes dropdown menu items to be displayed out of order.

Your answer here.

(b) (3 points)

A bug that causes some images to swap locations.

Your answer here.

Each of the following questions (c)-(e) gives a pair of concepts. It can be a pair of techniques, a pair of tools, or a pair of processes, etc. Consider the reading *"Analyze This! 145 Questions for Data Scientists in Software Engineering"*. For each pair, choose a quoted respondent question from Section 4.1 of that reading, copy the exact quote into the answer space, and then argue that one element of the pair would be better at answering that quoted question than the other.

For example, given the pair **Code Inspection vs. Measurement**, you might copy the quote "How long does it take to gain a payoff from moving to new software-writing tools?" and then argue that Measurement is more relevant to answering that question.

For each question, after copying the quote, please use at most four sentences. Use each quote at most once.

(c) (3 points)

A/B Testing vs. Risk

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

(d) (3 points)

Regression Testing vs. Integration Testing

Your answer here.

(e) (3 points)

Fuzz Testing vs. the Eraser Dynamic Analysis

Your answer here.

(f) (6 points)

Suppose you are asked the following question during a technical interview (the particular programming language isn't relevant for this problem): **"Write a method that takes a singly-linked list as input and reverses that list."** Write 2 questions you can ask of the interviewer before you start implementing the solution. For each, use a direct quote from the reading *"The Google Technical Interview"* to support your answer. Copy the exact quote into the answer space as part of your answer. (Do not use a question that literally occurs in the reading, such as "How big could the input be?". Instead, create a new question and support it with a general claim from the reading.)

Your answer here.

You are an engineering manager who must decide whether or not to employ pair programming for a series of tasks (questions (g)-(k)).

Use the same sort of mathematical reasoning generally described in slides 18 and 19 of the lecture. For each task, write the pair programming cost divided by the solo programming cost (e.g., if the pair cost is 91 and the solo cost is 100, write 0.91). Use two figures after the decimal point (e.g., 1.23). The interpretation of the various features (e.g., "fewer total lines" or "fewer total defects") is as covered in class.

(g) (1 points)

Task 1: 50,000 LOC program, Coding at 25 LOC/hour, Defect rate of 10 defects / KLOC, Defect fix time of 20 hours / defect, Pair programming results in 15% fewer total defects, Pair programming results in 15% fewer total lines of code.

Your answer here.

(h) (1 points)

Task 2: 50,000 LOC program, Coding at 25 LOC/hour, Defect rate of 10 defects / KLOC, Defect fix time of 20 hours / defect, Pair programming results in 20% fewer total defects, Pair programming results in 10% fewer total lines of code.

Your answer here.

LATE

minutes remaining

Hide Time

(i) (1 points)

Task 3: 75,000 LOC program, Coding at 50 LOC/hour, Defect rate of 10 defects / KLOC, Defect fix time of 5 hours / defect, Pair programming results in 10% fewer total defects, Pair programming results in 12% fewer total lines of code.

Manual Save

Your answer here.

Navigation

Question 4. Mutation Testing (20 points)

Consider the following code.

```
1  int func2(int n) {
2    if (n <= 1) {
3      return n;
4    }
5    int a = 1;
6    int b = 4;
7    for (int i = 1; i < n; ++i)
8    {
9      int c = b - (2 * a);
10     a = b;
11     b = c;
12   }
13   return b;
```

(a-1) (3 points) Your mutants can swap between the operators <, == and <= (but no other operators). Your mutants can swap between the numbers 1, 0 and -1 (but no other numbers). If you are only allowed to mutate the base (line 2), how many mutants are killed by the test case with input n = 3?

Your answer here.

(a-2) (3 points) Your mutants can swap between the operators <, == and <= (but no other operators). Your mutants cannot change anything else. If you are only allowed to mutate the loop guard (line 7), how many mutants are killed by the test case with input n = 2?

Your answer here.

(b) (4 points)

Suppose that a group of researchers find very convincing evidence in favor of the competent programmer hypothesis but also find very convincing evidence against the coupling effect hypothesis. What would this mean for mutation testing? For credit, include a direct quote from *An Analysis and Survey of the Development of Mutation Testing* (part of the HW3 spec) to support your argument. (After copying the quote, use at most four sentences.)

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

(c-1) (10 points)

Consider the following Python function `distracted_goldstine(n)`:

```
1  def distracted_goldstine(n):
2      if n < 0:
3          return False
4      low, high = 0, n
5      while low <= high:
6          mid = (low + high) // 2
7          mid_squared = mid * mid
8
9          if mid_squared == n:
10             return True
11         elif mid_squared < n:
12             low = mid + 1
13         else:
```

We have exactly three test cases: -1, 1 and 16.

Your task is to create two different first-order mutants of the `distracted_goldstine(n)` function by modifying **only the branch conditions** (i.e., the expressions in `if`, `elif`, and `while` statements). Create your two mutants by editing the two copies of the code below. You can edit only the lines with the branch conditions; do not change other lines.

Each of your mutants should fail a different and non-zero number of these test cases. For example, your first mutant might fail one test, while your second mutant fails two tests.

Each mutant should have exactly one modification from the original code (i.e., each of your mutants should be a first-order mutant). For example, you may change comparison operators (e.g., `==` to `!=`, `<` to `<=`), logical operators, or negate conditions. Do not add or remove entire statements; focus only on modifying the conditions.

Below are two code boxes where you will create your mutants. Click on the lines with branch conditions (they are editable) to modify them. Remember to modify only the conditions in the `if`, `elif`, and `while` statements. You should not change other lines.

Mutant 1:

```
1  def distracted_goldstine(n):
2      if n < 0:
3          return False
4      low, high = 0, n
5      while low <= high:
6          mid = (low + high) // 2
7          mid_squared = mid * mid
8
9          if mid_squared == n:
10             return True
11         elif mid_squared < n:
12             low = mid + 1
13         else:
```

Mutant 2:

```
1  def distracted_goldstine(n):
2      if n < 0:
3          return False
4      low, high = 0, n
5      while low <= high:
6          mid = (low + high) // 2
7          mid_squared = mid * mid
8
9          if mid_squared == n:
10             return True
```

```
        elif mid_squared < n:
            low = mid + 1
        else:
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

Question 5. Dynamic Analysis (17 points)

You have developed a dynamic analysis tool, called Marbles, that aims to quickly identify potential memory leaks. A *memory leak* occurs when memory is allocated but never deallocated. In one run of a program, every allocation at address *x* must be paired with a deallocation at exactly address *x*. Failing to do so indicates a memory leak.

Marbles works by tracking and logging memory allocations and deallocations during program execution. Because dynamic analyses can be slow, you design Marbles so that it runs and analyzes multiple different threads of the same subject program code simultaneously. Each concurrent execution is assigned a unique thread number (e.g., Thread 1, Thread 2, etc.). Marbles collects a single unified log of information about each memory operation, potentially including the allocated memory address, deallocated memory address, size, and the thread number.

However, the Marbles instrumentation may be buggy, so the log may be missing events or details or may contain spurious events. Marbles reports memory leaks based on analyzing its log file, so this may result in false positives and/or false negatives.

In this context:
A *false positive* occurs when Marbles incorrectly reports the presence of a memory leak, but no leak is actually possible (on any execution).
A *false negative* occurs when Marbles fails to report a memory leak, but a leak is actually possible based on the source code (on some executions).

Consider the programs below. We have run Marbles on each of them. The log file for each analysis is also shown below. For each program, you are asked to determine if Marbles would report a memory leak based on the Marbles log file. In addition, you will also be asked to determine whether Marbles has incurred any false positives/negatives during the analysis process.

NOTE: You may need to scroll down on some of the code snippets to view the full program and/or log file.

```
    *ptr1 = 42;
    *ptr2 = 84;
    delete ptr1;
}

Marbles log file:
Thread 1: Allocate 4 bytes at address 0x7000
Thread 2: Allocate 4 bytes at address 0x7100
Thread 2: Allocate 4 bytes at address 0x7200
Thread 2: Deallocate bytes at address 0x7100
Thread 1: Allocate 4 bytes at address 0x7300
Thread 1: Deallocate bytes at address 0x7000
```

(a-1) (1 points) **True / False**: Marbles would report a memory leak for program `p1()`, based on the log file.

- ◯ True
- ◯ False

(a-2) (1 points) **True / False**: Marbles incurred a false positive or false negative for `p1`.

- ◯ True
- ◯ False

```
void p2() {
    // Dynamically allocates a new buffer (150 bytes)
    char* x = new char(150);
```

```
    if (foo) {
      char* y = &x[16];
    }
    delete[] x;
}
```

```
Marbles log file:
Thread 1: Allocate 150 bytes at address 0x8000
Thread 2: Allocate 150 bytes at address 0x8200
Thread 1: Deallocate bytes at address 0x8000
```

(b-1) (1 points) **True / False**: Marbles would report a memory leak for program `p2()`, based on the log file (the allocation at 0x8210 is not paired with a deallocation).

- ◯ True
- ◯ False

(b-2) (1 points) **True / False**: Marbles incurred a false positive or false negative for `p2`, during its logging process.

- ◯ True
- ◯ False

```
void p3() {
    // Dynamically allocate arr1 (32 bytes)
    int* arr1 = new int[8];
    // Dynamically allocate arr2 (12 bytes)
    int* arr2 = new int[3];
    // Deallocate arr2
    delete[] arr2;
    // Deallocate arr1
    if (random(0,100) <= 50) {
      delete[] arr1;
    }
}
```

(c-1) (1 points) **True / False**: Marbles would report a memory leak for program `p3()`, based on the log file.

- ◯ True
- ◯ False

(c-2) (1 points) **True / False**: Marbles incurred a false positive or false negative for `p3`, during the logging process.

- ◯ True
- ◯ False

```
void p4(){
    // Dynamically allocates arr1 (40 bytes)
    int* arr1 = new int[10];
        // Dynamically allocates arr2 (24 bytes)
    int* arr2 = new int[6];
    // Deallocates arr2
    delete[] arr2;
    // Deallocate arr1
    delete[] arr1;
}
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

(d-1) (1 points) **True / False**: Marbles would report a memory leak for program `p4()`, based on the log file.

- ◯ True
- ◯ False

(d-2) (1 points) **True / False**: Marbles incurred a false positive or false negative for `p4`, during the logging process.

- ◯ True
- ◯ False

(e) (3 points)

Compare and contrast two approaches for race condition detection: the dynamic analysis tool Eraser and static code inspection. Describe one situation in which the former would work well and the latter would not, then describe a situation in which the latter would work well and the former would not. Reference some of the human factors associated with code inspection from the lectures and reading. (Use at most six sentences.)

Your answer here.

(f) (3 points)

You are developing a real-time trading system that must process market data and be able to execute trades with low latency. Describe how dynamic analysis can help ensure the performance and the reliability of your system. What is the relationship between dynamic analysis utility and test suite coverage? (Use at most 3 sentences.)

Your answer here.

(g) (3 points)

In the lectures we discussed an implantable medical device and an associated bug. In that context, describe a situation in which a static analysis technique could be a better approach than a dynamic analysis technique to ensure security. (Use at most three sentences.)

Your answer here.

Extra Credit

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

Your answer here.

---

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

Your answer here.

LATE

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)

Your answer here.

(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)

Your answer here.

(5) 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)

Your answer here.

Honor Pledge and Exam Submission

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

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

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.