Navigation

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

For each statement below, input the letter of the term that is *best* described. 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. — Dynamic Analysis	G. — Fault Localization	H. — Formal Code Inspection
I. — Fuzz Testing	J. — Integration Testing	K. — Milestone	L. — Mocking
M. — Oracle	N. — Pair Programming	O. — Passaround Code Review	P. — Perverse Incentives
Q. — Race Condition	R. — Regression Testing	S. — Risk	T. — Sampling Bias
U. — Software Metric	V. — Static Analysis	W. — Streetlight Effect	X. — Triage
Y. — Unit Testing	Z. — Waterfall Model		

Q1.1: J

Kamilla is creating a new social network software together with their co-workers. Each of the functions written by Kamilla works as expected, but when put together with the other functions implemented by Kamilla's co-workers, the entire system does not work. Kamilla plans to test the entire software, with both Kamilla's functions and functions written by other people.

Q1.2: V

Engineers at Boomit use a tool, called DetectBugs, to find potential bugs and runtime errors in Boomit's codebase. In particular, the DetectBugs tool does not run Boomit's code.

Q1.3: L

Kameron wants to test some functionality of their code. The code involves using a method foo that takes a very long time to run. To speed up the testing, Kameron uses a way to replace foo with another implementation bar which takes significantly less time to run.

Q1.4: Z

To manage their new project, Daphne decides to execute the following steps in order: (1) gathering requirements of the software they want to develop, (2) designing and implementing the code, and (3) testing the system and fixing bugs.

Q1.5: H

The software engineers at Bioplex have a day-long meeting to go over their codebase together and discuss potential bugs and security risks.

Q1.6: S

There may be potential problems and unfortunate events that may compromise the success of projects. Experienced project leads will create strategies to mitigate such potential problems and control unfortunate event outcomes.

Q1.7: D

Before Kookle launches the brand new maps to their adventure video game, they send a version of the maps to some end-users for their feedback.

Q1.8: E

90% of the unit tests currently failed. However, after changing an `or` sign (||) to an `and` sign (&&), all the unit tests for the codebase passed.

Q1.9: G

Donware, an online marketplace company, is experiencing segmentation faults in their massive codebase. Henry identifies that memory reads are rarely the problem, but memory writes are. They write

a script to list lines of code that write to memory that come before memory reads that have caused segmentation faults.

Q1.10: O

After Paulina finishes their code for a new feature, they must get the changes looked over by another engineer before the changes are merged to BuzzyFuzzy's codebase. The other engineer could look at the changes offline, without having to meet with Paulina synchronously.

Q1.11: Q

Three concurrent threads, A, B, and C, access the same shared variable v without locking: all of them read the value of v, and thread A also writes to v. Depending how the threads interleave with each other, thread B may read a different value of v when the program is executed multiple times.

Q1.12: K

Kyle is a manager for a team of software engineers. To ensure that the team stays on track, Kyle sets an internal deadline to have 75% of the features completed for their prototype. This will not be seen by customers.

Q1.13: N

Dave and Ariel are classmates working on a coding project together. They find that when Dave types the code, and Ariel simultaneously watches to identify bugs and make suggestions, they are more effective than individually coding without synchronized communication with each other.

Q1.14: P

Gogozoom calculates their engineers' year-end bonuses by the number of lines of code the engineers write. This leads to engineers writing more lines of code that may be unnecessary and/or unreadable.

Question 2. Code Coverage (19 points)

You are given the following functions. Assume that statement coverage applies only to statements marked STMT_#, and we consider all statements marked STMT_# in the entire program when calculating coverage. That is, even if the program execution starts from one particular method, we consider coverage with respect to the contents of all methods shown. Similarly, even if some methods are not executed during the program execution, we consider coverage with respect to the contents of all methods shown. Finally, we assume every argument of string type has at least one character; in other words, we do not consider NULL or empty strings as input to functions in this program.

```
2     STMT_1
3     if (a[0] == b[0]) {
4         jelly_bean(strlen(a), strlen(b))
5     }
6     STMT_2
7 }
8
9 void jelly_bean(int x, int y) {
10    STMT_3
11    if (x < y) {
12        STMT_4
13        polar_bear(y)
14    } else {
15        STMT_5
16        x = 0
17        polar_bear(x)
18    }
19 }
20
21 void polar_bear(int z) {
22    while (z > 5) { // line 22
23        STMT_6
24        z--
25    }
26    if (z <= 2) { // line 26
27        STMT_7
28    }
29    STMT_8
30 }
31
```

Navigation

Navigation

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

(a) (3 points)

Provide **1** input (i.e., both arguments) to `maang(str a, str b)` such that the *statement* coverage under this input is **75%**. Write your test input in the form such as `maang(hello, world)`, if it is possible to achieve the given statement coverage. If it is not possible, enter "not possible".

In the context of this question, you have to pick inputs from the following seven strings: `{ av, att, meta, apple, amazon, nvidia, netflix }`.

Your answer here.

ANSWER: ('att', 'amazon') or (‘‘amazon’', '‘apple’') or any other input where the first letters match and (the first string is longer than the second string) or (the second string has >5 letters and is longer than the first string)

Different students were presented with different coverage targets. Some example answers include:

- 25: ('meta', 'amazon')
- 62.5: ('av', 'att')
- 75: ('att', 'amazon')
- 50: not possible

(b) (3 points) **True / False**: there exists a test suite (with at least one test input) such that the test suite obtains **100%** *statement* coverage. (We only consider statements marked **STMT_#** when computing statement coverage in this question. And we consider all statements marked **STMT_#**.) Furthermore, in this question you can use any test inputs, not necessarily only those strings from the previous question.

- ◯ True
- ◯ False

ANSWER: True A possible such test suite is {('apple', 'att'), ('att', 'amazon')}

(c) (3 points) What is the maximum branch coverage achievable by **one and only one** (i.e., exactly one) input to `maang(str a, str b)`? Note that, the `polar_bear` function has a while loop in it – similar to if-then-else, this while loop also introduces two branches: one branch that enters the loop body and the other branch exits the loop.

Your answer here.

ANSWER: 62.5%

The answer should be 62.5%. First, we can see that each while/if has 2 branches, meaning that the total number of branches to visit is 8. For each if/else, one single test case can only possibly reach one of the two branches. For the while in polar_bear(), on the other hand, one single test can manage to execute both branches. Thus, the maximum number of branches that may be visited in one pass is 1+1+1+2 = 5. 5/8 = 62.5%

(d) (3 points) What would be the maximum *statement* coverage achievable by **exactly one** input if the "while" condition on line 22 were replaced with z > 2?

Your answer here.

ANSWER: 87.5%

(e) (4 points) What is the minimum number of test cases to reach 100% *branch* coverage? Provide the test cases with their input in the form `maang(a, b)`. For a and b, please use string values from only the following seven strings: `{ av, att, meta, apple, amazon, nvidia, netflix }`. For example, one test case could be `maang(att, att)`. Another example test case is `maang(att, meta)`. Please write each test case on a separate line.

Your answer here.

ANSWER: At least three test cases are needed to reach 100% branch coverage. E.g., maang(att, meta) maang(apple, att) maang(att, amazon)

(f) (3 points) In 4 sentences or less, describe a scenario in which 100% statement coverage might miss a bug in a program.

Your answer here.

ANSWER: Answers will vary. In the lecture, examples such as division by zero and SQL injection were given. In a division by zero bug, you can visit the line with a non-zero denominator value and not see the bug. One way to find such an issue would be to use a dataflow analysis that determines if values are zero. Another example might be a race condition: you might have 100%

statement coverage but not observe the right scheduler interleavings. A tool such as Eraser or CHESS could help find the race condition in such a situation. Student responses should not exceed 4 sentences.

Navigation

Question 3. Short Answer (5 points each, 40 points)

(a) (5 points)

Suppose you are interviewing at company Corp481, and you get the following technical question:

Given an array of strings called `strs`, group the anagrams together. Here, an "anagram" is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. For example, “ate” and “eat” are anagrams.

What are three questions you may want to ask -- e.g., to help clarify the question, or to help you better understand the task, or to convince the interviewer that you understand relevant software engineering concepts -- before you start typing any code for this question?

Your answer here.

ANSWER:

Possible answers (non-exhaustive list):

a. What letters do strs[i] consist of?
b. Does order matter in the returned anagrams?
c. What is the expected data structure to store the results?
d. What is the maximum number of strings in strs?
e. What is the maximum number of characters in each strs[i]?
f. Can strs be empty?
g. Can strs[i] be empty?

(b) (5 points)

Suppose you are managing a team of software engineers at company Corp481.

After looking through the history of commits, you realized that each individual code change is quite large. You decided to encourage developers on your team to keep each individual change small going forward.

How would you justify this decision of breaking larger changes into a series of smaller changes? Feel free to cite what you learned from the lecture slides and/or readings to back up your justifications. Please use at most five sentences.

Your answer here.

ANSWER:

It's easier for code review (From the Henderson’s reading: “Engineers are encouraged to keep each individual change small, with larger changes preferably broken into a series of smaller changes that a reviewer can easily review in one go." This also makes it easier for the author to respond to major changes suggested during the review of each piece; very large changes are often too rigid and resist reviewer-suggested changes”).

Note: there may be other reasons, and the response should be credited accordingly.

(c) (5 points)

Suppose you are managing a team of software engineers at company Corp481.

When developing a large project, different components in the project may take different amounts of time to be implemented. You find that programmers who finish their work early are oftentimes blocked by the work of other programmers. For example, programmer A cannot proceed to test her function because it requires the output of the function which programmer B is currently still working on.

In order to improve the overall efficiency of the entire team, what **single** Software Engineering method can you apply **and** why is it a good choice? Please use at most five sentences and include at least two reasons why your method is a good choice.

Navigation

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

Your answer here.

ANSWER:

Mocking. When the full implementation of a method is not yet available, mocking can reduce the degree of dependencies and allow simultaneous development rather than sequential.

(d) (5 points)

Suppose you are managing a team of software engineers at company Corp481.

In order to improve productivity, you plan to base developer end-of-year cash bonuses on the following metrics:

a. The number of words of documentation written.
b. The number of code changes accepted during code reviews.

Evaluate the pros and cons of each of these two metrics. Use less than 2 sentences for pros and less than 2 sentences for cons.

Your answer here.

ANSWER:

Note: there may be other reasons, and the response should be credited accordingly.

a. (1)The amount of documentations written: makes documentations unnecessarily verbose and long
b. (2)The number of merge requests accepted during code reviews: this encourages splitting a meaningful merge/pull request into trivial, small ones. this is also unfair to those developers who are doing tasks that don't require creating many merge requests.

You need to decide whether or not to employ pair programming (i.e., two programmers code up the task together) for a series of tasks. You will only opt in for pair programming if it leads to an overall lower cost ($$). Otherwise, you would choose to use individual programming (i.e., one developer programs the entire task **alone**).

Suppose for ALL tasks, pair programming makes coding 20% slower but results in 60% fewer defects. For example, a task — that takes one programmer 10 hours to complete — would take a pair of two programmers 12 hours to complete (i.e., two programmers are pair programming together for the entire 12 hours). On the other hand, given a task, suppose one programmer writes a program to solve this task that has 10 bugs. If two programmers pair programs together, they would write a program that solves the same task and that has 4 bugs.

In the context of this question, when pair programming, we allow two programmers to write the program together, however, when fixing bugs/defects, each programmer will do it individually. In other words, in terms of fixing defects, there is no difference between pair programming and individual programming: a defect is always fixed by one programmer.

The hourly rate for each programmer would be $50. That is, if a task takes one individual programmer 10 hours to code up, the cost is $500 (i.e., we need to pay the programmer $500). On the other hand, if two programmers pair program for 10 hours, the total cost would be $1,000 (i.e., each programmer gets paid $500). As for fixing defects, if a defect takes one programmer 1 hour to fix, the programmer would get paid $50.

The following tables detail the specifications for each task. In particular, for each task, it gives:

a. Program Size (LOC): the total lines of code (LOC). Note that, for the purpose of this question, pair programming and individual programming will produce programs of the same size.
b. Coding Speed (LOC / hour): the number of lines of code per hour that one programmer can write for the task.
c. Defect Rate (#defects / KLOC): the number of defects produced per one thousand lines of code, assuming one programmer is working on the task alone.
d. Defect Fixing Rate (#hours / defect): the number of hours for one programmer to fix one defect.

In each answer box, enter either "Individual" or "Pair" as your answer

---

(e) (5 points)

Program Size (LOC)	Coding Speed (LOC / hour)	Defect Rate (#defects / KLOC)	Defect Fixing Rate (#hours / defect)
100,000	50	20	10

Your answer here.

Navigation

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

ANSWER:

As individuals: 2000 hr coding + 20000 hr fixing defects = 22000 hr As pairs: 2400 hr coding + 8000 hr fixing defects = 10400 hr Cost for individuals = 22000 hr * 50 = $1,100,000 Cost for pairs = 10,400 hr * 100 = $1,040,000 Answer: pair

---

(f) (5 points)

Program Size (LOC)	Coding Speed (LOC / hour)	Defect Rate (#defects / KLOC)	Defect Fixing Rate (#hours / defect)
100,000	100	10	2

Your answer here.

ANSWER:

As individuals: 1000 hr coding + 2000 hr fixing defects = 3000 hr As pairs: 1200 hr coding + 400 hr fixing defects = 1600 hr Cost for individuals = 3000 hr * 50 = $150,000 Cost for pairs = 1600 hr * 100 = $160,000 Answer: individual

---

(g) (5 points) You are working on a multi-threaded C++ codebase with many *lock* and *unlocks*. In every function or method, there are many *if* statements that check for errors that result in an early *return*, many of which are obscure and expected to almost never happen. You just learned that you need to call *unlock* before all *return* statements, and the only reason you've not yet run into any issues is because you only forgot to *unlock* in some of these error checks. You want to use dynamic analysis to identify inputs that cause these problems. Is this a good technique, or is there a better one for this scenario? In addition, please indicate which dynamic analysis from the lecture or readings you think would be the 'best' fit for this situation. Justify your answer. Limit your answer to no more than five sentences.

Your answer here.

ANSWER:

Static analysis should be preferred since whether a lock is unlocked can be traced at every `if-return`. Dynamic analysis can be justified, but static analysis is preferred because errors that induce these early returns are rare so dynamic analysis is unlikely to find them, and even if they do, dynamic analysis tools for lock/unlock raise many false positives.

(h) (5 points) You are a software engineer at an app-based rideshare company with a very large codebase. Because of a recent high-profile hack at one of your competitors, you and your coworkers decide to systematically evaluate your codebase for security vulnerabilities. If your main concern is identifying whether a defect exists that would cause employee credentials to be leaked, what single method would you use to evaluate the quality of your codebase, and why? If you have multiple methods in mind, please explain a best one in your answer. Use no more than five sentences.

Your answer here.

ANSWER:

Probably dynamic analysis, since the program can be instrumented to record every part of the program that reads the credentials variables. It also avoids the efficiency limitation because the main concern is employee credentials, so only a subset of the codebase needs to be tested. Static analysis is also an acceptable answer with sufficient justification, but is impractical because variables are read very often.

Question 4. Mutation Testing (8 points)

Consider the following python program that implements a standard binary search algorithm. Note that the '//' operator is the floor division operator in python. For example, `9 // 2` gives you 4. We are interested in the two mutants that are created for this python program — they are shown in the comments below. In particular, mutant 1 changes to "high = len(arr)" without changing anything else. Mutant 2 changes only the "while" condition from using "<=" to using "<".

```
# Returns index of x in given array arr if present,
# else returns -1

```

Navigation

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

```python
def binary_search(arr, x):
    high = len(arr) - 1    # mutant 1: high = len(arr)
    mid = 0
    low = 0

    while low <= high:    # mutant 2: while low < high
        mid = (high + low) // 2

        if arr[mid] < x:
            low = mid + 1
        elif arr[mid] > x:
            high = mid - 1
        else:
            return mid

    return -1
```

(a) (4 points)

Your first task is to fill in the following table. In this table, you are given two test inputs. For example, in the first test input arr = [1, 2, 3, 4] and x = 2 and in the second test input arr = [1, 2, 3, 4] and x = 1. Indicate in the table below whether or not each of these test inputs would kill each mutant.

Test #	Input	Oracle	Mutant 1 Killed?	Mutant 2 Killed?
1	arr = [1, 2, 3, 4] x = 2	1	◯ True ◯ False ANSWER: False	◯ True ◯ False ANSWER: False
2	arr = [1, 2, 3, 4] x = 1	0	◯ True ◯ False ANSWER: False	◯ True ◯ False ANSWER: True

(b) (4 points) In a few sentences, discuss the pros and cons of using each of the two mutants to evaluate the each of the two test inputs

Your answer here.

ANSWER: Demonstrates a clear understanding of the principles of mutation testing AND applies them to the given tests and mutants accurately

Question 5. Invariants (8 points)

Consider the following code snippet that defines a function, called `totallyUsefulFunction`, which takes as input two non-negative integers and one boolean value. In particular, the `apple` variable is a non-negative integer (i.e., positive or zero) that indicates the number of apples. Similarly, `banana` is the number of bananas (positive or zero, but not negative). Finally, `chocolate` variable is a boolean that is either True or False, meaning whether or not we have chocolate.

The function calculates a particular `foodScore`. Each apple contributes 1 point and each banana contributes 2 points. If chocolate is not present, the total `foodScore` is always 0.

The function also calculates the `foodCount` – the total number of apples, bananas and chocolate (with True counting as one item and False as zero items).

```
totallyUsefulFunction(int apple, int banana, bool chocolate){

    int foodScore = apple + banana * 2;
    int foodCount = apple + banana + 1;

    while (foodScore < 22){
        apple ++;
        foodScore ++;
        foodCount ++;
    }

    // Can I have a little chocolate, as a treat?
    if (chocolate == false){
        foodScore = 0;
        foodCount --;
        print('sad');
    } else {
        print('yum!')
```

```
19      }
20
21      // Invariants Evaluated Here
22
23  }
```

(a) (8 points)

Consider the following four invariants generated by an oracle for `totallyUsefulFunction`. (The oracle here could be an automated dynamic invariant generation tool like Daikon. Or it could be created manually by a human. How this oracle is implemented is not important in this question. We just assume we're given four invariants.)

```
INV_1: foodCount >= 10
INV_2: foodCount >= 12
INV_3: foodCount == apple + banana + 1
INV_4: foodScore >= 0

```

The invariants are evaluated at the end of the function – as indicated at the end of the `totallyUsefulFunction`. For each invariant, please first indicate either (1) the invariant is valid, or (2) the invariant is not valid.

Then, explain your reasoning. That if, for each invariant, if it is valid, please briefly explain why you believe it is valid. If invalid, please describe a situation in which this invariant is violated. For example, you could specify the parameter values that make the invariant invalid – please use the format like `[apple = 0, banana = 0, chocolate = false]`.

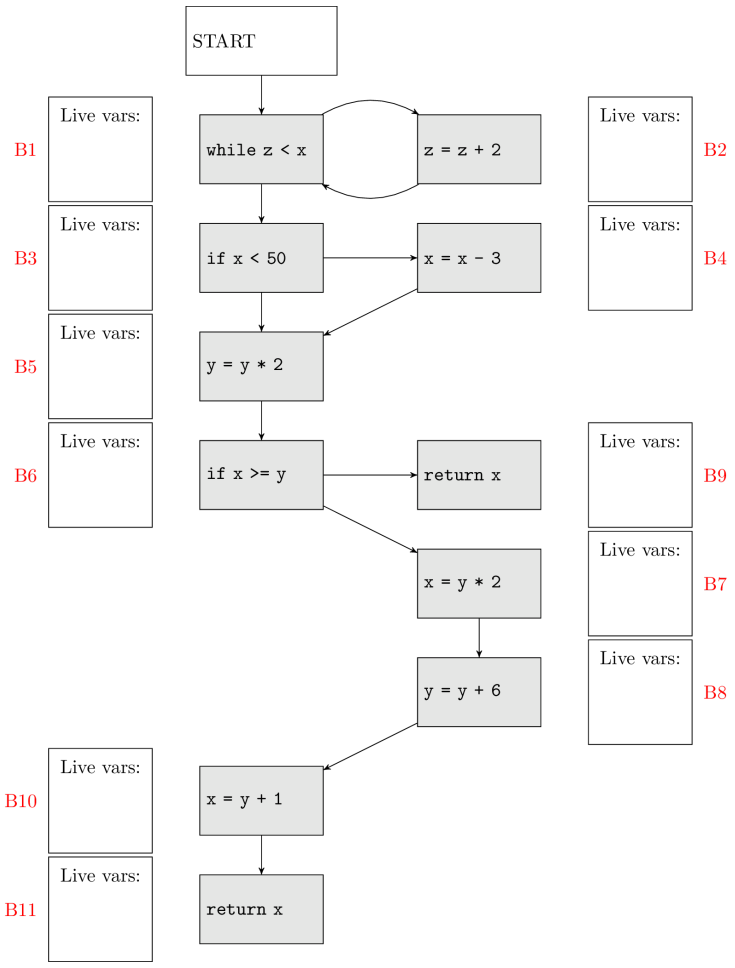
Please provide your answer for each invariant in the box below. Each invariant is a new line. You can use the format like `INV_1: valid-or-invalid. your-reasoning.`

Your answer here.

ANSWER: Answers will vary, grade by hand

Question 6: Dataflow Analysis (11 points total)

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.)



Navigation

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

B2
ANSWER: {'y', 'z', 'x'}

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

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

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

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

B7
ANSWER: {'y'}

B8
ANSWER: {'y'}

B9
ANSWER: {'x'}

B10
ANSWER: {'y'}

B11
ANSWER: {'x'}

Extra Credit

Each question below is for 1 point of extra credit unless noted otherwise. We are strict about giving points for these answers. No partial credit.

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

Your answer here.

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

Your answer here.

(3) If you read any optional reading, identify it and demonstrate to us that you have read it. (2 points)

Your answer here.

(4) If you read any *other* optional reading, identify it and demonstrate to us that you have read it. (2 points)

Your answer here.

(5) In your own words, identify and explain any of the bonus psychology effects. (2 points)

Your answer here.