You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

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 cell to mark it off. Each word is used at most once.

A. — A/B Testing	B. — Alpha Testing	C. — Beta Testing	D. — Competent Programmer Hypothesis
E. — Coupling Effect Hypothesis	F. — Feature Request	G. — Formal Code Inspection	H. — Fuzz Testing
I. — Instrumentation	J. — Integration Testing	K. — Invariant	L. — Milestone
M. — Mocking	N. — Pair Programming	O. — Passaround Code Review	P. — Perverse Incentive
Q. — Race Condition	R. — Regression Test	S. — Severity	T. — Spiral Development
U. — Streetlight Effect	V. — Test-driven Development	W. — Traceability	X. — Triage
Y. — Unit Testing	Z. — Waterfall Model		

Q1.1:

Taki is writing Virtual Reality software for Mitsuha. They agree upon a set of measurable outcomes for the software that will be delivered at the end of each year in their three-year contract.

Q1.2:

After finishing a prototype, teams at Whoosh Video Conferencing provide the prototype to an internal Quality Assurance team to identify early issues with the prototype.

Q1.3:

Before a proposed change can be merged into the main branch, at least 2 other developers need to look over and sign off on the changes. This process happens without much preparation; once a change is ready to be reviewed, it can be done so at developers leisure.

Q1.4:

Tenshi is writing a voice-activated app called *HandSonic* that supports three services: Delay, Harmonics, and Distortion. To measure how long it takes each service to execute, Tenshi adds code to measure the start and end time of each service.

Q1.5:

Kevin has recently finished creating a chat bot that answers questions about philosophy. Regardless of the question, the chatbot always answers "Only time will tell".

Q1.6:

Garry is writing a multithreaded game in which multiple threads access the shared screen buffer concurrently. Garry might use a tool like Eraser to avoid *this*.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

Q1.7:

The software development team at Watchflix develops new products by delivering increasingly-complex prototypes of the desired product every four months.

Q1.8:

Before starting to work on a new feature, Luigi creates tests that maps out the expected behavior of the new feature. This way, they will be able to tell if their implementation meets specifications.

Q1.9:

During code review, Chase realizes that they forgot to write tests to assert that a key requirement is being satisfied. To ensure this does not happen again, they annotate each test case with a specific requirement.

Q1.10:

Mizuki is developing a multiplayer networked factory simulation game called Saddestfactory. The test the multiplayer functionality, Mizuki writes a script to randomly generate network packets fed as input to the game to expose potential defects in handling network activity.

Q1.11:

ForkKnife's product uses a publicly-available Maps API in production. However, for testing purposes, the return values of the API calls are hard-coded.

Q1.12:

After creating major changes to the code base, Mohona notices that previous issues that they had resolved are showing up again. They resolve these issues and add tests such that, in the future, these issues will not arise again without alerting the developer.

Q1.13:

Muge is debugging her potato-boiling app. Leveraging *this* insight, she writes a test suite comprised of simplistic test cases, knowing that these simple test cases will also find more complex faults.

Q1.14:

Moondew Mountain's software has two components: a single-threaded program and a multi-threaded program. Moondew Mountain's testers find more bugs in the single-threaded program than in the multi-threaded program because it is easier to write tests for the single-threaded program.

Question 2: Coverage (20 points total)

You are given the following C functions. For simplicity, assume that `bool` means an integer 0 for False, and 1 for True. Further assume that statement coverage applies only to statements marked `STMT_#`; ignore all other statements from consideration when computing coverage.

```
void Blandroid(int a, int b, int c) {
    if (wake(a, b)) {
        return;
    }
    wave(a, b, c);
}

int wake(int a, int b) {
    if (a != b) {
        STMT_1;
    } else if (a != b) {
        return 0;
```

You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

Q2.1 (2 points) Suppose you can choose one set of inputs (A, B, C) for the function invocation `Blandroid(A, B, C)`. True or False: *it is possible to select one set of inputs (A, B, C) to achieve 20% or greater statement coverage over the code above.* Assume that covering all of `STMT_1` through `STMT_5` counts as 100%.

○ True ○ False

Q2.2 (5 points) Provide values for inputs `A`, `B`, and `C` that achieves the *highest* statement coverage possible for this file by executing `Blandroid(A, B, C)`. Again, assume we only count coverage of `STMT_1` through `STMT_5`.

A

A

B

B

C

C

Q2.3 (3 points) Support or refute: the test input you provided above achieves the lowest possible path coverage for the given code snippet.

Support or refute: the test input you provided above achieves the lowest possible path coverage for the given code snippet.

Q2.4 (10 points total, 1 point per selection) Next, consider the code below. Make selections for the operators `P`, `Q`, `R`, `S`, and `T` such that:

1. The number of *paths* in the function `focused_orangutan` is maximized.
2. The *path coverage* induced by executing the single test case `focused_orangutan(0, 1, 0)` is maximized.

```
void focused_orangutan (bool a, bool b, bool c) {
    if       (a   •P•    b) STMT_1;
    else if  (b   •Q•    c) STMT_2;
    else      STMT_3;

    STMT_4;

    if       (b    •R•    c) STMT_5;

    STMT_6;

    if       (b    •S•    c) STMT_7;
    else if  (a    •T•    b) STMT_8;
```

Operator	Maximize Number of Paths	Maximize Path Coverage
P	○ == ○ < ○ >	○ == ○ < ○ >
Q	○ == ○ < ○ >	○ == ○ < ○ >
R	○ == ○ < ○ >	○ == ○ < ○ >
S	○ == ○ < ○ >	○ == ○ < ○ >

Operator	Maximize Number of Paths	Maximize Path Coverage
T	○ == ○ < ○ >	○ == ○ < ○ >

You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

Question 3: Short Answer and Potpourri (28 points total)

Provide answers to each question below.

(6 points) Q3.1 Dynamic analysis and instrumentation.

Suppose you instrument a program to collect how long each function takes to run. The program contains 3 functions: `foo`, `bar`, and `qux`. Your instrumentation *only* records the average time elapsed between the start and end of every function. Assume for this question that the program executes `foo` exactly one time (i.e., that `main` immediately calls `foo`).

Running your instrumented program provides the following information for one run of the program:

Cumulative Time for `foo` (ms)	Cumulative Time for `bar` (ms)	Cumulative Time for `qux` (ms)
103	3	20

Based on the information above, explain the order in which `foo` could have invoked `bar` or `qux` to produce the data above. Multiple answers may apply. Use three sentences or fewer. Assume functions are not recursively called. *Hint: `qux` may call `bar` or vice versa; `qux` and `bar` may call each other multiple times.*

For example, if you believe that `foo` calls `bar` two times, and that `bar` calls `qux` once, you could explain how that sequence of function calls produces the data observed in the table above. There is no specific syntax here — do your best to explain and justify the order in which the functions could be called to produce the data above.

Your answer here.

Q3.2 (2 points each; 6 points total)

Read the scenario statements below. For each statement, choose the best combination of Task, Goal, and Target by taking one selection from each column in the table below. For example, `a 1 p` is valid, but not `a b 2` or `c 3`. If multiple combinations might fit, choose the *best* answer, or the one corresponding to a tool or technique from the class or the readings. You may use an option more than once across multiple scenario statements.

For example, given the statement: *You use a tool to enumerate thread interleavings in your multithreaded program to create tests that expose race conditions.* then you might answer "Automated dynamic analysis", "Generate", "Unit tests" if you think that scenario is best addressed by applying an automated instrumentation such as CHESS to generate test cases that reveal concurrency issues.

(Q3.2.1) You read through your code and see that a semi-colon is missing.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

Task:

- (a): Manual static analysis
- (b): Manual dynamic analysis
- (c): Automated dynamic analysis
- (d): Software engineering process
- (e): Automated static analysis

Goal:

- (1): Minimize
- (2): Transform
- (3): Maximize
- (4): Generate
- (5): Validate

Target:

- (m): Unit tests
- (n): Abstract syntax tree
- (o): Program source code
- (p): Control flow graph
- (q): Integration tests

(Q3.2.2) You use a compiler that examines program structure to report syntax errors.
Task:

- (a): Manual static analysis
- (b): Manual dynamic analysis
- (c): Automated dynamic analysis
- (d): Software engineering process
- (e): Automated static analysis

Goal:

- (1): Minimize
- (2): Transform
- (3): Maximize
- (4): Generate
- (5): Validate

Target:

- (m): Unit tests
- (n): Abstract syntax tree
- (o): Program source code
- (p): Control flow graph
- (q): Integration tests

(Q3.2.3) You use a tool such as AFL to help improve the quality of your test suite.
Task:

- (a): Manual static analysis
- (b): Manual dynamic analysis
- (c): Automated dynamic analysis
- (d): Software engineering process
- (e): Automated static analysis

Goal:

Q3.3 (5 points per correct row, 10 points total) Consider the following code examples taken from a binary tree implementation with containing defects. Of the options shown, explain what the defect is, or indicate `None` if no defect exists. Then, select which QA method(s) is/are most likely to reveal the defect or ensure the code works as intended.

Code snippet 1

```
def height (root):
    # Returns the height of a binary tree

if root.right:
    height
else:
    return root.value


```

You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

The code above may have a defect. It could be an entirely wrong implementation (e.g., it computes something different from what the comment indicates), or it could be a single syntactic problem (e.g., missing a symbol). Describe what you think the defect is. If you do not believe a defect is present, mark the space with None.

Descibe the defect, if present. Otherwise, indicate "None"

Now, consider each QA approach below. Mark the option(s) that you think would reveal the defect (or leave them blank if you think no defect exists).

- ☐ 100% Coverage
- ☐ EvoSuite
- ☐ Pex-like
- ☐ Pair Programming
- ☐ Formal Code Inspection

Code snippet 2

```
def search(root, value):
    if root.value == value:
        return true
    elif root.val != value:
        return search(root.right, value)
    else:
        return search(root.left, value)
```

The code above may have a defect. It could be an entirely wrong implementation (e.g., it computes something different from what the comment indicates), or it could be a single syntactic problem (e.g., missing a symbol). Describe what you think the defect is. If you do not believe a defect is present, mark the space with None.

Descibe the defect, if present. Otherwise, indicate "None"

Now, consider each QA approach below. Mark the option(s) that you think would reveal the defect (or leave them blank if you think no defect exists).

- ☐ 100% Coverage
- ☐ EvoSuite
- ☐ Pex-like
- ☐ Pair Programming
- ☐ Formal Code Inspection

Q3.4 (2 points each; 6 points total) Pair programming; process

You are a manager at WebFlix and need to decide whether or not to employ pair programming for a series of tasks. Since pair programming tends to produce code of higher quality, you are willing to opt for pair programming for a particular task so long as there is not an increase in total costs of more than 6%. The table below summarizes the various costs and benefits of using pair programming for each task.

Task	Total Hours	Cost Per Hour	Pair Programming decrease in Total Hours (%)	Pair Programming increase in Cost per Hour (%)
A	38	5	51	38
B	36	15	71	12
C	21	10	42	103

You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

(2 points each) For each of the following tasks, decide whether to employ pair programming.

A
○ Yes, use Pair Programming ○ No, do not use Pair Programming

B
○ Yes, use Pair Programming ○ No, do not use Pair Programming

C
○ Yes, use Pair Programming ○ No, do not use Pair Programming

Question 4. Mutation Testing (17 points)

Consider the code snippet below defining a function `foo`:

```python
def foo(a, b, c):
    bar = -1
    if (a >= 3):
        bar += 4
    elif (c < 11 and 9 > c):
        bar -= 4
    if (c >= b or c == 0):
        if (a == 0):
            bar = 0
        else:
            bar = bar + 4 * (a // 5)
    if (a == 3 or c == 9):
        bar = 15
```

Given test input `(0,14,20)`, the function produces an output of `0`.

(a) (4 points per mutant, 12 total) Make at most one edit each to create three mutants of foo such that the kill score of the suite of three mutants, when provided the same test input, is 2/3.

You should not introduce any loops as part of your mutations. Make sure that your mutants correspond to valid Python3 code — syntactically invalid mutants may receive no credit. Moreover, please do not attempt to subvert this question by modifying the code to immediately return a value — you are asked to make single-order mutants.

Attempting to submit code that infinitely loops, that interacts with any I/O, that imports other libraries, or that shells out is a violation of the honor code. Doing so will result in a 0 for the entire exam.

Mutant 1:

```python
def foo(a, b, c):
    bar = -1
    if (a >= 3):
        bar += 4
    elif (c < 11 and 9 > c):
        bar -= 4
    if (c >= b or c == 0):
        if (a == 0):
            bar = 0
        else:
            bar = bar + 4 * (a // 5)
    if (a == 3 or c == 9):
        bar = 15
```

Mutant 2:

```python
def foo(a, b, c):
    bar = -1
    if (a >= 3):
        bar += 4
    elif (c < 11 and 9 > c):
        bar -= 4
    if (c >= b or c == 0):
        if (a == 0):
            bar = 0
```

You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

```
        else:
            bar = bar + 4 * (a // 5)
    if (a == 3 or c == 9):
        bar = 15
```

Mutant 3:

```
def foo(a, b, c):
    bar = -1
    if (a >= 3):
        bar += 4
    elif (c < 11 and 9 > c):
        bar -= 4
    if (c >= b or c == 0):
        if (a == 0):
            bar = 0
        else:
            bar = bar + 4 * (a // 5)
    if (a == 3 or c == 9):
        bar = 15
```

(5 points) Support or refute in four sentences or fewer: It is easier to kill higher order mutants than it is to kill single order mutants, therefore mutation testing suites should focus on higher order mutations.

Write your answer here.

Question 5. Mutation Analysis and Invariants (10 points)

You are given the following snippet of C++ code:

```
int foo(int a, int b, bool c) {
    int x = a + b;
    int y = a * b;
    int z = 0;
    for(int i = 0; i < a*b; i += a) {
        if ( i == a && c) {
            continue;
        }
        z++;
        x += a + b;
        y += b;
      //Evaluate invariants here
    }
```

An oracle generates the following invariants for the snippet:

```
INV_1: (z <= (b-1)) || !c || i != a
INV_2: !c || (i==a)
INV_3: i < a
```

The invariants are evaluated at the end of each iteration BEFORE the loop increment. For each invariant, provide a set of inputs that violates the invariant or indicate that the invariant is valid. The range of the integer inputs is restricted to positive integers (`> 0`). Booleans can be `true` or `false` (please note they are all lowercase — it will help our grading). Express your answer as a Python dictionary.

For example, if you believe that `INV_1` is valid and that `INV_2` can be invalidated with inputs `[1, 3, true]`, then you would write: `{"INV_1": true, "INV_2": [1, 3, true]}` (again, note the brackets for expressing the inputs as a list).

You are viewing kjleach

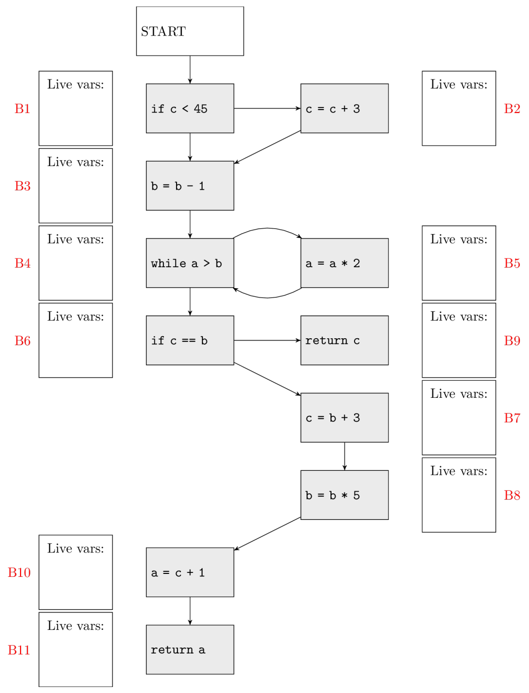
Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

Question 6: Dataflow Analysis (11 points total)

Consider a *live variable dataflow analysis* for three variables, a, b, and c used in the graph below. We associate with each variable a separate analysis fact: either the variable is possibly read on a later path before it is overwritten (live) or 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 {a, b}. The special statement return reads, but does not write, its argument. (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., ab to indicate that a and b are alive before that block.

B1		B2		B3		B4	
B5		B6		B7		B8	
B9		B10		B11			

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.

You are viewing kjleach

Prev: (none)
Next: hammada.

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Extra Credit
- Pledge & Submit

(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.

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.*

Ignore me