124

minutes remaining

Hide Time

Manual Save

Navigation

Question 1. Word Bank Matching (1 point each, 16 points total)

For each statement below, input the letter of the term that is **best and most specifically** described. Note that you can click each cell to mark it off. Each word can only be used at most once.

Sometimes multiple words may appear to apply. Pick the best and most specific match. As one example, a hypothetical prompt about "running a program on an input and using a comparator to check the output against an oracle" logically matches both "testing" and "quality assurance". In such a case you would want to pick "testing", because "testing" is more specifically described.

A. — Code Review	B. — Delta Debugging	C. — Design Patterns	D. — Elicitation
E. — Fault Localization	F. — Informal Goal	G. — Maintainability	H. — Priority
I. — Productivity	J. — Profiling	K. — Readability	L. — Requirements
M. — Risk	N. — Severity	O. — Stakeholder	P. — Static Analysis
Q. — Triage	R. — Validation	S. — Watchpoint	T. — Weak Conflict

Q1.1: Detailed descriptions of what the software/program should do.

Q1.2: Before changed code is accepted, other developers must inspect the proposed change and its justification. This iterative process is mandated at most major software engineering companies.

Q1.3: The team meets to discuss and assign a priority to each one of a list of reported defects.

Q1.4: You are reading the project specification for Project 2 in EECS 370. You think something seems wrong with the specification itself, so ask the course member about a possible mistake in the specification.

Q1.5: When debugging, you are interested in stopping the program's execution every time the global variable `global_request_hash` changes. Since that variable is assigned to in many potential places, you employ ...

Q1.6: The potential negative effect of a bug on the coding or running of a system, encompassing what happens if the bug is not fixed.

Q1.7: Product managers at Spotify meet with artists, record companies and lawyers to find out what new feature they want the most. What role do artists, record companies and lawyers play in this scenario?

Q1.8: This static quality property of software relates to its ability to be comprehended and thus maintained.

Q1.9: You are working on Euchre for EECS 280. You know there is a bug in your `player.cpp` file because you are failing a relevant test case. Now you are trying to figure out which specific lines are likely implicated in this bug.

Q1.10: You analyze your code for projects in 281, measuring the space and time complexity. You want to know the values of these quality properties so that you can optimize them.

Q1.11: A high-level notion in requirements elicitation that is communicated but is not precise enough to be measured.

Q1.12: You are working at a database company. In conversations, the client requests that "temporary tables must be deleted within five weeks" and then also requests that "temporary tables must be retained long enough for regulatory compliance".

Q1.13: You are the product manager at TikTok. You are given a new task to work on a new feature on the backend of the application. You begin by trying to better understand what it should do by communicating with the stakeholders.

Q1.14: These are best practices that are employed to solve commonly-recurring problems across multiple pieces of software. Common solutions often relate to object creation, structure or behavior.

Q1.15: An examination of the potential behaviors of a program without actually running it. May be manual or automatic and is often conservative.

Q1.16: You are working at Accenture at the start of the pandemic. It is possible that many of your developers may fall ill and may thus not be able to complete their tasks on time. You set up a meeting to discuss this possibility, its consequences, and its potential mitigations.

Question 2. Delta Debugging (21 points)

(a) (2 points each; 6 points total) Consider the three problems below. For each problem, specify if delta debugging can be used to solve the problem. If it can, provide a brief description of an "Interesting" function that will help solve the problem. If it cannot, specify which properties of delta debugging make it not suitable to solve the problem.

(i) (2 points) We have a list of distinct strings, and every letter of the alphabet is present at least once somewhere in at least one of the strings in the list. We want to find a one-minimal subset of this list such that every letter of the alphabet is still present at least once somewhere in at least one of the strings in the subset.

Your answer here.

(ii) (2 points) In an effort to reduce the memory footprint of our C++ program, we have decided to replace all uses of `int` with `short`. However, when we do so, our program fails some of its test cases (we suspect due to integer overflow). We want to identify which `int` fields can be replaced with `short` fields such that the program still passes all tests.

Your answer here.

(iii) (2 points) We have a list of distinct non-negative integers. We want to find a one-minimal subset of this list of integers such that the sum of the subset is greater than twenty.

Your answer here.

(b) We decide to design a new version of the delta debugging algorithm, shuffle debugging, which doesn't require any set intersections or unions. Instead, if neither the first half nor the second half of the set is interesting, shuffle debugging finds an irrelevant element (if one exists), performs a riffle shuffle of the two halves (in which the two halves are interleaved element by element, also called an "in shuffle"), and tries over again. If no irrelevant element exists, we conclude that we have a one-minimal set. We redesign the algorithm as described in the following Python code (note - make sure to read the full code if some is cut off):

```
1
2  # Interleave lists A and B, return a new list containing
3  # alternating elements of A and B every other.
4  # Ex:
5  #   interleave([1, 2, 3], [4, 5, 6]) = [1, 4, 2, 5, 3, 6]
6  #   interleave([], [4, 5, 6]) = [4, 5, 6]
7  #   interleave([1, 2], [3, 4, 5, 6]) = [1, 3, 2, 4, 5, 6]
8  def interleave(A, B):
9      result = []
10     for i in range(max(len(A), len(B))):
11         if i < len(A):
12             result.append(A[i])
13         if i < len(B):
14             result.append(B[i])
15     return result
16
17 # Split a list in half and return (first_half, second_half).
```

```
18 # If the list isn't evenly divisible, the length of the first
19 # half will be smaller than that of the second half
20 def split(l):
21     first_half = l[:len(l)//2]
22     second_half = l[len(l)//2:]
23     return first_half, second_half
24
25 def SD(C):
26     if len(C) == 1:
27         return C
28     # P1 is the first half, and P2 is the second
29     P1, P2 = split(C)
30     if interesting(P1):
31         return SD(P1)
32     if interesting(P2):
33         return SD(P2)
34     for i in range(len(C)):
35         # C_prime is every element of C except the one at index i
36         C_prime = [x for j, x in enumerate(C) if j != i]
37         if interesting(C_prime):
38             # C_i isn't necessary, so remove it
39             P1_prime, P2_prime = split(C_prime)
40             shuffled = interleave(P1_prime, P2_prime)
41             return SD(shuffled)
42     # The set is already one-minimal
43     return C
44
45
```

We test out this new algorithm on the following example:
`Interesting(X) = [1, 9] is a subset of X`
(2 points each; 6 points total) For each of the following values of `C`, list how many calls to `Interesting` are made when `SD(C)` is called. If the algorithm never terminates, write `INCONCLUSIVE`.

(i) (2 points) `C = [1, 2, 5, 0, 4, 3]`

Your answer here.

(ii) (2 points) `C = [1, 4, 3, 2, 5, 0, 9]`

Your answer here.

(iii) (2 points) `C = [1, 9, 5, 3, 4, 2, 0]`

Your answer here.

(c) (3 points each; 9 points total) In order for shuffle debugging (SD) to find a one-minimal subset, some of the requirements of delta debugging may or may not be necessary for shuffle debugging. For each of the following requirements, indicate whether violating the requirement could result in SD failing to return a one-minimal subset ("necessary") or whether SD can always operate correctly on inputs that do not meet that requirement ("optional"). If a requirement is necessary for shuffle debugging, give an example where not having it results in an incorrect answer. If a requirement is optional for shuffle debugging, give an example where delta debugging would give the wrong answer but shuffle debugging would give the right answer.

(i) (3 points) Monotonicity?

Your answer here.

(ii) (3 points) Unambiguity?

Your answer here.

(iii) (3 points) Consistency?

Your answer here.

Question 3. Short Answer (18 points)

Provide an answer to each of the questions below.

(a) (3 points) In 3 sentences or less, describe and explain three things that we can do to reinforce the readability of our code.

Your answer here.

(b) (3 points) In 4 sentences or less, describe and explain the tradeoffs/differences between working in academia and working in industry according to Dr. Ciera Jaspan and/or Dr. Kevin Leach.

Your answer here.

(c) (3 points) In 2 sentences or less, describe and explain two major benefits of using Medical Imaging to gather information about software engineering processes.

Your answer here.

(d) (3 points) In 3 sentences or less, describe two key differences between experts and novices with respect to problem solving and productivity.

Your answer here.

(e) (3 points) In 1 sentence, give an example of a quality requirement. In 2 sentences or less, describe and explain the connections between the environment and the machine.

Your answer here.

(f) (3 points) Consider a game of Nim with four piles: {A,A,A,A,A}, {B,B}, {C,C,C}, {D,D,D,D,D,D,D} (i.e., 5 2 3 7). It is your turn. What move would you take to win?

Your answer here.

Question 4. Fault Localization (14 points total)

Below is a buggy snippet from a Python program that determines which of the 3 values passed in is the largest (and returns 0 if they are all equivalent):

```
          ...
1   class differentValues(object):
2     def greatestValue(self, a, b, c):
3       if (a > b || a > c):
4         return a
```

```
5       else if (b > a || b > c):
6         return b
7       else if (c > a && c > b):
8         return c
9       return 0
           ...
```

Use the standard Tarantula Fault-Localization Suspiciousness Ranking for necessary calculations:

```
Sus(s) = (failed(s) / totalfailed) / (failed(s) / totalfailed + passed(s) / totalpassed)
```

... where totalpassed is the number of passing test cases and passed(s) is the number of those that executed line s (similarly for totalfailed and failed(s)). Note that this is the same formula presented in the Lecture: there is no "trick" in the formula.

(a) (6 points) Give two sets of inputs that cause line 5 to have the highest suspiciousness ranking overall. The first set should pass (i.e., the program obtains the correct answer) and the second set should fail (i.e., the current program obtains the incorrect answer). Use only the values 1, 2 and 3 (in any combination, with repeats if you like) in your answer. Format your answer with the values as triplets by commas and place one answer on each line, as in:

```
(4,5,6)
(7,8,9)
```

Your answer here.

(b) (4 points) Choose either **library-oriented architectures** (one approach to designing for maintainability) or **multi-language projects** (focusing on the glue code that interfaces between two languages). Identify your choice, and then either **support** or **refute** the claim that Tarantula-style fault localization would be effective at localizing defects in that context. Use at most four sentences.

Your answer here.

(c) (4 points) Give an example of a security defect, attack or exploit for which we would expect Tarantula-style fault localization to do a good job of pinpointing the right line to change. Then given an example of a security defect, attack or exploit for which we would expect it to do a poor job. Use at most four sentences total (e.g., example, explanation, example, explanation).

Your answer here.

Question 5. Design Patterns (17 points total)

Consider the following C++ code that will serialize the value of one particular node in a Binary Search Tree (BST). This code snippet is only relevant for parts 5a, 5b and 5c. You may assume the following:

- `parseArgs` will correctly parse any given arguments. Arguments may include a pointer to the root node of the tree as well as the node to find
- All nodes may have a unique integer value (i.e., `node->val`)
- Aside from `serializeNode`, `parseArgs`, and `serializeIntHelper`, you may assume there are no other functions and interfaces
- All functions not defined here are implemented correctly

```
1
2  string serializeNode(void* args) {
3      auto argsPar = parseArgs(args);
4      Node* root = argsPar["root"];
5      Node* nodeToFind = argsPar["input"];
6
7      if (root == nullptr) {
8          throw runtime_error("Error");
9      }
10
11     // Perform Breadth-First Search
12     queue<Node*> bfs;
13     bool nodeFound = false;
```

124

minutes remaining

Hide Time

Manual Save

Navigation

```
14      int nodeVal = -1;
15      bfs.push_back(root);
16
17      while (!bfs.empty()) {
18          Node* curr ;
19
20          // Loop invariant is true here
21
22          curr = bfs.front();
23          bfs.pop_front();
24
25          if (curr == nodeToFind) {
26              nodeFound = true;
27              nodeVal = curr->val;
28              goto nodeFound;
29          }
30
31          if (curr->left != nullptr) {
32              bfs.push_back(curr->left);
33          }
34
35          if (curr->right != nullptr) {
36              bfs.push_back(curr->right);
37          }
38      }
39
40  nodeFound:
41
42      if (!nodeFound) {
43          throw runtime_error("Error");
44      } else {
45          // Serialize the node
46          string serializedVal = serializeIntHelper(nodeVal);
47          return serializedVal;
48      }
49  }
50
51
```

5a. (2 points) List one invariant of the function. The invariant should be true inside the loop at the line indicated by the comment. Do not restate any of the assumptions listed above. Do not list a predicate that is trivial or true for all programs (e.g., `1+1=2`).

Your answer here.

5b. (2 points) List one post-condition of the function. The post-condition should be true as the function terminates (either via a `return` or an uncaught exception). Do not restate any of the assumptions listed above. Do not list a predicate that is trivial or true for all programs (e.g., `1+1=2`).

Your answer here.

5c. (3 points) Describe why duplicate code can have a negative impact on software maintenance. Describe how you might refactor the original code to improve maintainability. Use at most four sentences total.

Your answer here.

5d. (6 points) Consider **Singleton**, **Template Method**, and **Strategy** design patterns. For each design pattern, provide two specific examples of programs that may utilize the design pattern and explain why such a program benefits from this design pattern. Use at most four sentences per design pattern (context, benefit, context, benefit).

Your answer here.

5e. (2 points) Suppose that a particular software engineering project spends 39% of its lifetime effort on implementation, 39% of its lifetime effort on testing, and 22% of its lifetime effort on other non-testing maintenance. You are considering a design that would (a) increase the effort required for implementation by 14% (for example, if implementation previously took 10 hours, but that effort is increased by 35%, it would now take 13.5 hours); but that would also (b) reduce the effort required for testing by 14%. Assume the project originally required 100 effort units to complete over its lifetime: calculate the new lifetime effort units required by the project with your new proposed design.

Your answer here.

5f. (2 points) In four sentences or fewer, reference the paper *The art of software systems development: Reliability, Availability, Maintainability, Performance (RAMP)* and other knowledge to support or refute the following claim: We **always** want to spend more on design to *reduce* maintenance costs. You should include at least two pieces of evidence or argument, at least one of which should be associated with the paper. Use at most four sentences.

Your answer here.

Question 6. Interviews (14 points total)

A candidate at a technical interview is given the following prompt:

Write `isAnagram()`, a function that returns whether two strings are anagrams or not. String A and string B are anagrams if A's characters can be rearranged to form string B.

Below, the candidate provides an implementation for isAnagram():

```
bool isAnagram(string A, string B) {
    // creates a size 26 array filled with zeroes
    int charCounter[26];

    for (char letter : A) {
        charCounter[letter - 'a']++;
    }

    for (char letter : B) {
        --charCounter[letter - 'a'];
        if (charCounter[letter - 'a'] < 0) {
            return false;
        }
    }

    for (int count : charCounter) {
        if (count != 0) {
            return false;
        }
    }

    return true;

}


```

6a. (1 points) Identify a test input that would return true for the above implementation.

Your answer here.

6b. (1 points) Identify a test input that would return false for the above implementation.

Your answer here.

6c. (4 points) The interviewer runs the implementation on a set of test cases. 6 test cases pass, while 4 fail. The interviewer suggests that perhaps the candidate made some assumptions that resulted in an erroneous implementation. List two questions that this candidate could have asked such that each question would have revealed a separate mistaken assumption. (For example, candidates implementing integer division might avoid a divide-by-zero error by asking "can the denominator be zero?".)

Your answer here.

6d. (4 points) Assume that the company makes hiring decisions solely based on this programming interview task: candidates are evaluated based on the questions they ask during the interview and the performance of their submitted code on test cases. Identify and explain two flaws in this hiring process with respect to the most common company hiring goals (e.g., not hiring people who are likely to do a poor job, hiring people who are likely to do a good job, etc.).

Your answer here.

6e. (4 points) Support or refute the claim that *automated program repair* would likely repair the defect(s) in the code listed above, assuming a test suite with 100% line coverage that contains at least one test the code currently passes and one test the code currently fails.

Your answer here.

Question 7. Extra Credit (1 point each)

(*Feedback*) What was your favorite topic covered during the course?

What is one thing you like about this class?

(*Feedback*) What was your least favorite course topic (or the thing you would most recommend that we change for future semesters)?

What is one thing you dislike about this class?

(*Guest Lecture*) List one thing you learned from guest speaker Dr. Ciera Jaspan of Google or otherwise convince us that you paid careful attention during that lecture.

Jaspan guest lecture.

(*Optional Reading 1*) Identify a single optional reading that was assigned after Exam 1. Write a sentence about it that convinces us you read it critically.

Optional Reading 1

(*Optional Reading 2*) Identify a different single optional reading that was assigned after Exam 1 or a "long instructor post" that was posted after Exam 1. Write a sentence about it that convinces us you read it critically.

Optional Reading 2

(*Guest Lecture*) List one thing you learned from guest speaker Dr. Kevin Leach that was not listed on an introductory summary slide or otherwise convince us that you paid careful attention during that lecture.

Leach guest lecture

(*Optional Lectures*) List one thing you learned from a "Game Theory", "World Building", and/or "Quantum Computing and Romance Novels" lecture that was not listed on an introductory summary slide.

Optional bonus lectures

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.