Navigation

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

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.

A. — Adapter Design Pattern	B. — Bug Bounties	C. — Debugger	D. — Delta Debugging
E. — Fault Localization	F. — Flat Profile	G. — Functional Requirements	H. — Maintainability
I. — Named Constructor Idiom Pattern	J. — Profiler	K. — Quality Requirements	L. — Readability
M. — Requirements Elicitation	N. — Risk	O. — Singleton Design Pattern	P. — Stakeholder
Q. — Validation	R. — Verification	S. — Watchpoint	

Q1.1:

K

MemeStop is building a server stack that must be able to handle 1000 queries per second.

Q1.2:

C

Engineers at WeTube use a piece of software to single-step through their program a single line or instruction at a time. They can pinpoint trouble areas by monitoring variable values.

Q1.3:

O

Kuroko and Kazari are building software for a certain scientific railgun. Rather than using global variables, they track properties of the railgun within one instance of a class.

Q1.4:

H

Despite it taking twice as long to code, the engineers at Wintendo adopt the use of design patterns into their mobile app because it will reduce *this* required in the future.

Q1.5:

A

The C++ Standard Library implements the std::queue using the std::deque by default. This is an example of what?

Q1.6:

R

You run unit tests to ensure that your algorithmic trading bot is functioning correctly with respect to specifications.

Q1.7:

J

Kenneth is a systems validation engineer at Netskong. They use this tool to measure how many function calls are made and how long each take during runtime in their analysis report to their manager.

Q1.8:

B

General Vehicles rewards anonymous contributors with cash payments in exchange for demonstrations of its software's defects and vulnerabilities.

Navigation

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

Q1.9:

N

During a pitch for a new video game, Kai includes an assessment of the likelihood and consequences of failing to launch on time.

Q1.10:

L

During pass-around code review, your manager, Ms. PyLint, suggests breaking down a long Python list that is all on one line into separate lines, putting each list item on its own line.

Q1.11:

I

Rikka is writing software to manage all of her character's psychic powers in a Power class. She decides to implement multiple static public methods that each return new instances of the Power class.

Q1.12:

D

In addition to being a software developer, Yael is a world-renowned chef. They are creating a new paelle recipe, but must consider far too many ingredients. They decide to use an algorithm to efficiently determine which subset of ingredients are best for optimizing paella flavor. They call it Zaella.

Q1.13:

F

Hazel shares with you a performance report of their program, but it does not report the order or context in which functions are called. Instead, it reports only the average time spent in each function.

Q1.14:

G

Simon is trying to build a music app. The app must correctly identify the tempo and time signature of each song it plays.

Question 2: Debugging Delta Debugging (17 points)

You have a Delta Debugging algorithm identical to the one in lecture which is going wrong somewhere, and you have turned to every college student's favorite solution: print statement debugging! You know that there are 8 cases in the original, and that the set of interesting cases are [0, 5, 6]. There is at least one line of the pseudocode implementation given in lecture where a list intersect or a list subtraction is used in place of the list union which is supposed to be used. Your algorithm prints just before each recursive call and p and c at the beginning of each call. Your output is the following:

```
==========Top of the function==========
P is: []
c is: ['0', '1', '2', '3', '4', '5', '6', '7']
Interference
==========Top of the function==========
P is: ['4', '5', '6', '7']
c is: ['0', '1', '2', '3']
Interference
==========Top of the function==========
P is: ['2', '3', '4', '5', '6', '7']
c is: ['0', '1']
Interference
==========Top of the function==========
P is: ['1', '2', '3', '4', '5', '6', '7']
c is: ['0']
==========Top of the function==========
P is: ['0', '2', '3', '4', '5', '6', '7']
c is: ['1']
==========Top of the function==========
P is: ['0', '1', '4', '5', '6', '7']
c is: ['2', '3']
Interference
==========Top of the function==========
P is: ['0', '1', '3', '4', '5', '6', '7']
c is: ['2']
==========Top of the function==========
P is: ['0', '1', '2', '4', '5', '6', '7']
c is: ['3']
==========Top of the function==========
P is: ['0', '1', '2', '3']
c is: ['4', '5', '6', '7']
Interference
==========Top of the function==========
P is: ['0', '1', '2', '3', '6', '7']
c is: ['4', '5']
Interference
==========Top of the function==========
P is: ['0', '1', '2', '3', '5', '6', '7']
c is: ['4']
==========Top of the function==========
P is: ['0', '1', '2', '3', '4', '6', '7']
c is: ['5']
==========Top of the function==========
P is: ['0', '1', '2', '3', '4', '5']
c is: ['6', '7']
Interference
==========Top of the function==========
P is: ['0', '1', '2', '3', '4', '5', '7']
c is: ['6']
==========Top of the function==========
P is: ['0', '1', '2', '3', '4', '5', '6']
c is: ['7']
```

(9 points) In the box below, explain what evidence you found to support that conclusion.

Your answer here

(8 points) What is the issue with your Delta Debugging implementation? Assume you desire an implementation the matches that shown in lecture.

- ○ if Interesting(P ∪ P1)
- ○ if Interesting(P ∪ P2)
- ○ DD(P ∪ P2, P1)
- ○ DD(P ∪ P1, P2)
- ○ DD(P ∪ P2, P1) ∪ DD(P ∪ P1, P2)

Question 3: Short Answer (20 points total)

Provide answers to each question below.

Navigation

(3 points) Q3.1 Kristi Ramachandran.

In Kristi Ramachandran's Guest Lecture, she mentions that formal code reviews are used at APL. List a reason formal code inspection might be used over code reviews, in particular at APL.

Your answer here.

ANSWER: Flight software has high importance, formal code inspection is typically better at catching bugs; all or nothing points based on if argument is valid

(3 points) Q3.2 Kristi Ramachandran.

Explain in two sentences Kristi Ramachandran's perspective on agile/scrum development and how she uses it with her team at APL.

Your answer here.

ANSWER: uses scrum, not in the way a full scrum master would, need to tailor agile experience, informal sprint reviews, emphasis on time estimation and sprint planning; all or nothing points based on if argument is valid + information is accurate

(3 points) Q3.3 Fault Localization.

```
DStar = failed(s) / (passed(s) + (totalfailed - failed(s)))
```

Describe a bug where coverage based fault localization with the DStar algorithm would be an effective method to find the bug and explain why it would be effective. Use two sentences or fewer.

Your answer here.

ANSWER: bug should cause program crash frequently, extra exit(1) statement, use an undeclared variable, etc (could also give examples of code and make multiple choice); 1 pt for bug 1 pt for explanation

(4 points) Q3.4 Requirements.

In two sentences, describe a functional requirement and a quality requirement each for AFL and EvoSuite.

Your answer here.

ANSWER: 1 pt functional AFL, 1 pt quality AFL, 1 pt functional EvoSuite, 1 pt quality EvoSuite

(2 points) Q3.5 Brains.

In two sentences or fewer, describe how understanding brain activity associated with software engineering tasks can improve automated program repair.

Your answer here.

ANSWER: We mainly expect to see a discussion about informing readability models or other approaches to comprehension to improve acceptability or trustworthiness of patches produced by APR.

(2 points) Q3.6 Productivity.

Based on the productivity lecture, support or refute the claim that it is more efficient to buy slow, cheap development workstations for a large team of mediocre engineers in one sentence. Use two sentences or fewer.

Your answer here.

ANSWER: 1 pt for support/refute with valid reasoning

(3 points) Q3.7 Design Patterns

Consider a scenario in which you are tasked with developing an autonomous vehicle software system that integrates (1) sensor processing, (2) localization, (3) actuation, and (4) navigation components. All components are used in turn by a single Orchestrator. In autonomous vehicle systems, each component is implemented with a class and public methods used by the Orchestrator. In the space below, describe how you could use the publish-subscribe design pattern to implement this software system in two sentences or fewer. Then, describe how you could use the template design pattern in two sentences or fewer.

Your answer here.

ANSWER: Publish-Subscribe -- each component subscribes to the orchestrator, or the orchestrator subsribes to each component.
Template: orchestrator exposes an abstract method that each component implements.

Question 4. Automated Program Repair (18 points total)

Navigation

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

Each snippet below contains a defect (and that at least one test case is failing). Each comment indicates the desired correct behavior of the snippet.

For each snippet, indicate (1) whether or not the defect could likely be repaired by Automated Program Repair as discussed in lecture, and (2) why or why not.

Part (a):

```
// returns the floating point average of a vector of integers
double avg(const vector<int> &v) {
    double s = 0;
    for (int i = 0; i < v.size(); ++i) {
        s += v[i];
    }
    return (s / v.size());
}
```

(1 point) Yes/No: repairable by APR?

○ Yes, readily fixed by APR
○ No, not readily fixed by APR

(4 points) Next, Justify your answer in three sentences or fewer.

Justify your answer here.

Answers will vary. Generally, the `double avg` and `double range` functions *should* by fixable with APR because the types of defects were single keyword or operators that fall within the scope of AST modifications that are the basis for APR. However, the Pokemon-related code is *not* as readily repaired with APR. The issue is that the defects relate to quantities in string literals. These are generally semantics that (currently) require human expertise to make a judgment.

Part (b):

```
// returns the range of values within a vector of integers
long int range(const vector<long int> &v) {
    assert(v.size() >= 2);

    long int _min = INT_MAX, _max = INT_MIN;
    for (int i = 0; i < v.size(); ++i) {
        _max = max(_max, v[i]);
        _min = min(_min, v[i]);
    }
    return 1 + _max - _min;
}
```

(1 point) Yes/No: repairable by APR?

○ Yes, readily fixed by APR
○ No, not readily fixed by APR

(4 points) Next, Justify your answer in three sentences or fewer.

Navigation

Justify your answer here.

Answers will vary. Generally, the `double avg` and `double range` functions *should* by fixable with APR because the types of defects were single keyword or operators that fall within the scope of AST modifications that are the basis for APR. However, the Pokemon-related code is *not* as readily repaired with APR. The issue is that the defects relate to quantities in string literals. These are generally semantics that (currently) require human expertise to make a judgment.

Part (c):

```
class PsychicType { ... };
class Abra : public PsychicType { ... };
class Kadabra : public Abra { ... };
class Alakazam : public Kadabra { ... };


// this code should return
// a Abra object if the string pokemon equals "Abra",
// a Kadabra object if the string pokemon equals "Kadabra", or
// a Alakazam object if the string pokemon equals "Alakazam".
PsychicType* PsychicTypeFactory(const string &pokemon) {
    if (pokemon == "Barbara") {
        return new Abra;
```

(1 point) Yes/No: repairable by APR?

- ○ Yes, readily fixed by APR
- ○ No, not readily fixed by APR

(4 points) Next, Justify your answer in three sentences or fewer.

Justify your answer here.

Answers will vary. Generally, the `double avg` and `double range` functions *should* by fixable with APR because the types of defects were single keyword or operators that fall within the scope of AST modifications that are the basis for APR. However, the Pokemon-related code is *not* as readily repaired with APR. The issue is that the defects relate to quantities in string literals. These are generally semantics that (currently) require human expertise to make a judgment.

Part d:

(4 points) In five sentences or fewer, support or reject the claim that APR can by readily applied to assembly code. In particular, identify relevant issues or assumptions with APR and concretely indicate how those can or cannot be solved at the assembly level.

Justify your answer here.

Answers will vary here. APR should be able to apply regardless of the prompt you were given, just in slightly different ways. For example, Java code is ultimately a source code from which ASTs can be derived. On the other hand, APR on assembly code can work, but likely means more failing mutants (e.g., because there is less semantic information in a stream of assembly instructions).

Question 5. Fault Localization (16 points total)

The following snippet of code implements a method `lengthOfLongestSubstring` that, given an input string, finds the longest substring in the input string without repeating characters:

```
 1  class FunWithStrings(object):
 2  
 3      def lengthOfLongestSubstring(self, s: str) -> int:
 4          ec = dict()
 5          i, j = 0, 0
 6          mx = 0
 7          if len(s) == 0: return 0
 8          while j < len(s):
 9              if s[j] in ec:
10                  mx = max(mx, j-i)
11                  while s[i] != s[j]:
12                      del ec[s[i]]
13                      i -= 1
```

However, there is a bug in `lengthOfLongestSubstring`. You are trying to use the Tarantula fault localization metric to identify the buggy line.

(a) (1 point each; 6 total) Given the following four passing and failing test cases, specify which lines are covered and missed by each test case. *Separate line numbers using a semicolon (i.e., ";") as a delimiter.* The first row has been filled out for you.

Test Case	Test Result	Lines Covered	Lines Missed
`"abcabcbb"`	`fail`	1; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13	18; 19
`"bbbbb"`	`pass`	Covered lines	Missed lines
`"b"`	`pass`	Covered lines	Missed lines
`""`	`pass`	Covered lines	Missed lines

(b) (1 point per box; 10 total) The formula for the Tarantula fault localization metric is as follows:

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

Using information about lines visited for failing and passing tests, determine the top five most suspicious lines of code in the snippet, in descending order suspiciousness scores (i.e., most suspicious line first). Break ties in suspiciousness scores by asending line order (i.e., if lines `2` and `3` both have score `0.500`, then use `(2, 0.500)` and `(3, 0.500)` as your ordering. List the suspiciousness scores correct to 3 decimal places.

Line Number	Suspiciousness score
Most Sus Line	Sus Score
Second Most Sus Line	Sus Score
Third Most Sus Line	Sus Score
Fourth Most Sus Line	Sus Score

Line Number	Suspiciousness score
Fifth Most Sus Line	Sus Score

Navigation

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

ANSWER: Your unique answer is shown below.

```
hammada

abcabcbb,fail,[1; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15; 16; 17],[18; 19]
bbbbb,pass,[1; 3; 4; 5; 6; 7; 8; 9; 10; 11; 14; 15; 16; 17; 18; 19],[12; 13]
b,pass,[1; 3; 4; 5; 6; 7; 8; 9; 16; 17; 18; 19],[10; 11; 12; 13; 14; 15]
,pass,[1; 3; 4; 5; 6; 7],[8; 9; 10; 11; 12; 13; 14; 15; 16; 17; 18; 19]

12,1.000
13,1.000
10,0.750
11,0.750
14,0.750
```

Question 6: Profiling (15 points)

Consider the function call profile below. The `main` function is run with an *event-based profiler* and the following profile is generated:

Call graph profile:

- `1 * main()`
  - `1 * whizz()`
    - `5 * bar()`
    - `5 * piff()`
  - `1 * foo()`
    - `1 * bar()`
    - `1 * piff()`

Self-times for each function:

- main: 0 ms
- whizz: 60.0 ms
- foo: 120.0 ms
- bar: 240.0 ms
- piff: 240.0 ms

Note that the profile expresses the number of times each function was invoked in a specific context. That is, `main` is invoked once, and other functions may be invoked one or more times from there. The indentation in the list above corresponds to the context in which that function was called.

The *self-time* of a function is the time the function's stack frame spends on the top of the stack per invocation.

`main()` is then run with a *statistical profiler*. Assume that the self-times remain constant across runs. Write the names of each function in descending order of probability that a random probe of the profiler would interrupt the program in that function. Your answer must be a Python-formatted list. For example, if you believe that the order should be (most probable) `A, B, C, D` (least probable), you should answer: `["A", "B", "C", "D"]`. Your list should contain 5 elements.

Navigation

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

ANSWER: Your unique answer is shown below.

```
[('bar', 1440.0), ('piff', 1440.0), ('foo', 120.0), ('whizz', 60.0), ('main', 0)]
```

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 was your favorite topic in this course?

Your answer here.

(2) What was your least favorite topic in this course?

Your answer here.

(3) If you read any optional reading from the second half of the semester, identify it and demonstrate to us that you have read it. (2 points)

Your answer here.

(4) If you read any *other* optional reading from the second half of the semester, 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 discussed in the second half of the semester. (2 points)

Your answer here.