Practice Comprehensive Test Questions
CS422/622

1. Consider the following code:

```
/** calculates multiplication of a and b and returns the result
 * @param a
 * @param b
 * @return a * b
*/
public static int russianMultiplication (int a , int b) {
      int z = 0;
      while  (a != 0) {
            if (a % 2 != 0) {
                  z = z + b;
            }
            a = a / 2;
            b = b * 2;
      }
      return z ;
}
```

a) Write a set of test cases to achieve statement coverage. You do not need to write any code for this (i.e., only provide input values).
b) Add test cases to the above set of test cases (if needed) to achieve branch coverage. You do not need to write any code for this (i.e., only provide input values).
c) Add test cases to the above set of test cases (if needed) to achieve "loop boundary adequacy". You do not need to write any code for this (i.e., only provide input values).
d) Do any of the test case(s) you have written detect any fault(s)? If so, specify which one(s) that do so and what fault(s) they reveal.
e) Are there any fault(s) in this method? Can you write any extra test cases so that the above method does not correctly return the result of multiplication of its two parameters?

2. Considering the following table, answer the questions below:

Characteristic	Block 1	Block 2	Block 3	Block 4
Value 1	< 0	0	> 0	
Value 2	< 0	0	> 0	
Operation	+	-	*	/

a) Write tests to satisfy Each Choice Criterion.
b) Write tests to satisfy the Base Choice Criterion. Assume base choices are value 1 = '> 0', value 2 = '> 0', and Operation = '+'.
c) Write test cases to achieve Pairwise Coverage criterion.
d) How many test cases are required to achieve *All Combinations* coverage? Why?

3. Considering the following, answer questions a through d:

```
// Effects: If s1 or s2 is null throw NullPointerException
// else return a (non null) Set equal to the intersection
// of Sets s1 and s2
public Set intersection (Set s1, Set s2)


Characteristic: Validity of s1
    - s1 = null
    - s1 = {}
    - s1 has at least one element


Characteristic: Relation between s1 and s2
    - s1 and s2 represent the same set
    - s1 is a subset of s2
    - s2 is a subset of s1
    - s1 and s2 do not have any elements in common
```

a) Does the partition “Validity of s1” satisfy the completeness property? If not, give a value for s1 that does not fit in any block.

b) Does the partition “Relation between s1 and s2” satisfy the completeness property? If not, give a pair of values for s1 and s2 that does not fit in any block.

c) Does the partition “Relation between s1 and s2” satisfy the disjointness property? If not, give a pair of values for s1 and s2 that fits in more than one block.

d) If the “Base Choice” criterion were applied to the two partitions (exactly as written), how many test requirements would result?

4. Given predicate $p = a \leftrightarrow (b \wedge c)$, answer the following questions:

a) List all the clauses.
b) Compute (and simplify) the conditions under which each clause determines predicate p.
c) Write the complete truth table for the predicate. For each clause , you need a column and then you should have a separate column for the predicate.
d) List all pairs of rows from your table that satisfy General Active Clause Coverage (GACC) with respect to each clause.
e) List all pairs of rows from your table that satisfy Correlated Active Clause Coverage (CACC) with respect to each clause.*)*

5. Consider the following method:

```
public static int fun(int arr[])      {
    int min = -1;
    // Creates an empty hashset
    Set<Integer> myset = new HashSet<>();

    // Traverse the input array from right to left
    for (int i = arr.length - 1; i >= 0; i--) {
        if (myset.contains(arr[i]))
            min = i;
        else
            myset.add(arr[i]);
    }
    return min;
}
```

a) What does this function accomplish? Explain in a few sentences (try to be as precise and specific as possible).
b) Assume we replace the statement "`int min = -1;`" with "`int min = 0;`" to create mutant M1. Write a test case that *does not* kill M1 and write a test case that *does* kill M1.
c) Assume we replace the statement "`min = i;`" with "`min = i + 1;`" to create mutant M2. Write a test case that *does not* kill M2 and write a test case that *does* kill M2.
d) Mutate "`for (int i = arr.length - 1; i >= 0; i--)`" using a valid/standard mutation operator to produce an equivalent mutant.
e) Do you see any faults in `fun`? If so, write a test case (or test cases) to reveal the fault(s).

6. Under what circumstances is it useful/worthwhile to utilize mock-testing? Write down (and explain) at least three such circumstances.

7. Name three issues (i.e., drawbacks) of *debugging by logging*?

8. Name three limitations of coverage criteria.

9. The explicit goal of mutation testing is to provide some quantification on the overall quality of a given test suite (i.e., mutation score), which is not exactly the main goal of software testing (recall that the primary goal of software testing is to reveal real software faults). How and in what ways can one leverage mutation testing to reveal software faults? Explain as thoroughly and precisely as you can.

10. What is the biggest drawback of mutation testing? Explain.

11. Suppose that coverage criterion C1 subsumes coverage criterion C2. Further suppose that test set T1 satisfies C1 on program P and test set T2 satisfies C2, also on P. Which statement(s) is/are incorrect? (select all that apply)

a) T1 necessarily satisfies C2
b) If P contains a fault, and T2 reveals the fault, T1 does not necessarily also reveal the fault
c) If P contains a fault, and T1 reveals the fault, T2 necessarily also reveals the fault
d) T2 necessarily satisfies C1

12. Which statement(s) is/are true? (select all that apply)

a) Finding input values to reach a certain point of interest (i.e. a certain location) in the source code is a decidable problem. In other words, it is always possible to decide if there exists or not input values to reach a certain point in the code.
b) In general we can claim that higher code coverage means higher chances of finding faults
c) Expected output is a necessary part of a test case.
d) Test suite T1 achieves 90% branch coverage level on program P and test suite T2 achieves 100% branch coverage level on the very same program P. We can conclude that T1 is 10% less effective (i.e., less successful) in finding faults.

13. Consider the following code along with its specs (Javadoc comment):

```
    /**
     * Find last index of element y in array x
     *
     * @param x array to search
     * @param y value to look for
     * @return last index of y in x; -1 if absent
     * @throws NullPointerException if x is null
     */
    // test: x = [2, 3, 5]; y = 2; Expected = 0
    public static int findLast (int[] x, int y) {
        for (int i=x.length-1; i >= 0; i--) {
            if (x[i] == y) {
                return i;
            }
        }
        return -1;
    }
```

a) Write two properties for the above function.
b) Does any of the properties you wrote result in a failure revealing a fault in the code? If not, can you write a property that does so?
c) Are there any faults in the code? What is/are it/they? Write (a) fix(es).

14. Explain what the “probe the distribution” is when testing programs that use (pseudo)random generators.

15. What is documentation testing? Explain.

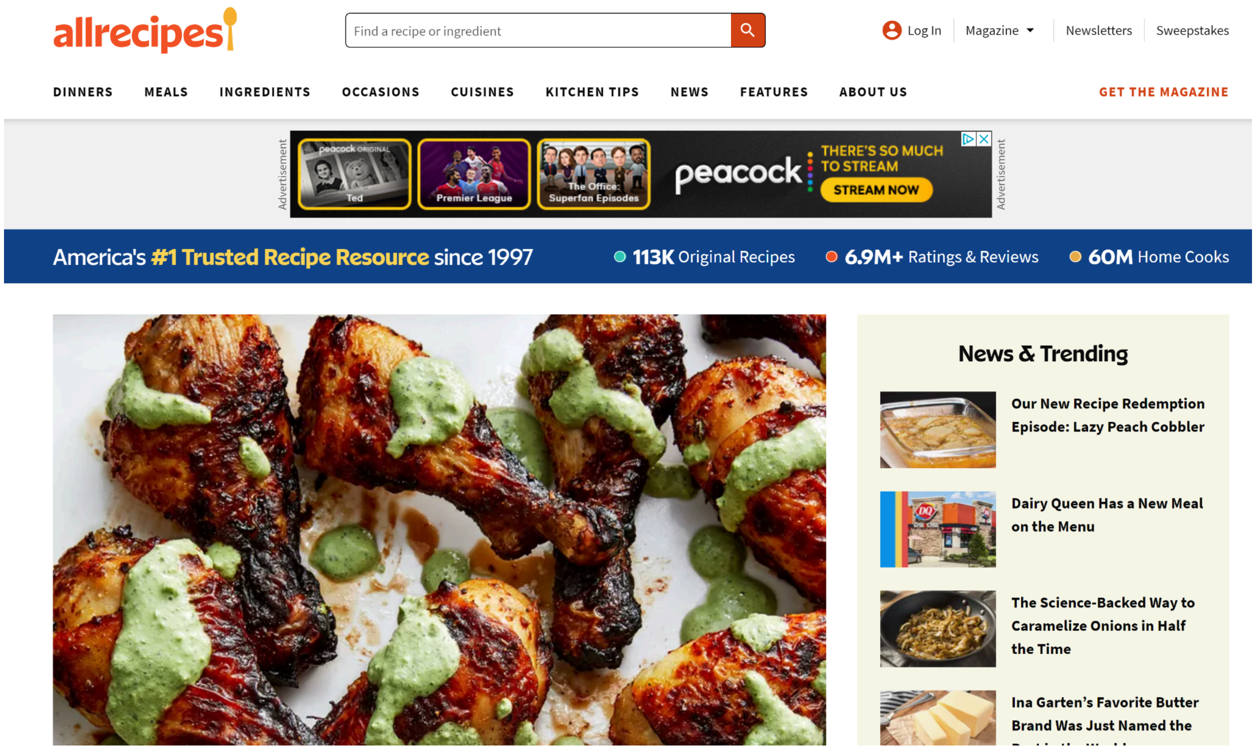
16. True/False?

i) We can be subjective in non-functional testing e.g., this GUI 1 is more user-friendly than GUI
ii) Documentation testing is a type of non-functional testing that deals with how easy-to-understand the user manuals/documentations of a software are.
iii) Load testing typically involves creating/utilizing virtual users to replay a test scenario against the app under test.
iv) Request per second is a measure in load testing that is equal to the total of number of requests that all virtual users collectively send to the system under test per second.

v) finding concrete inputs to falsify the implemented/formulated properties is part of Property-based Testing

17. What is the main difference between usability testing and A/B testing?

18. This is a snapshot of allrecipes.com. From a usability perspective, name at least 4 positive (advantages) of this design and at least one area of further improvement.



America's **#1 Trusted Recipe Resource** since 1997

113K Original Recipes · **6.9M+** Ratings & Reviews · **60M** Home Cooks

News & Trending

Our New Recipe Redemption Episode: Lazy Peach Cobbler

Dairy Queen Has a New Meal on the Menu

The Science-Backed Way to Caramelize Onions in Half the Time

Ina Garten's Favorite Butter Brand Was Just Named the