**Question 1**

**1 / 1 pts**

Which of the following statements is a valid objective for testing?



To prove that all possible defects are identified.



The test should start as late as possible so that development had enough time to create a good product.



To prove that any remaining defects will not cause any failures.

**Correct!**



To find as many failures as possible so that defects can be identified and corrected.

**Question 2**

**1 / 1 pts**

Which of the following statements correctly describes the difference between testing and debugging?



Testing removes defects; debugging identifies the causes of failures

**Correct!**



Dynamic testing shows failures caused by defects; debugging finds, analyzes, and removes the causes of failures in the software.



Dynamic testing prevents the causes of failures; debugging removes the failures



Testing identifies the source of defects; debugging analyzes the defects and proposes prevention activities.

**Question 3**

**0 / 1 pts**

Which one of the statements below describes the most common situation for a failure discovered  
during testing or in production?



The wrong version of a compiled source code file was included in the build.



The computation algorithm used the wrong input variables.

**Correct Answer**



The product crashed when the user selected an option in a dialog box.

**You Answered**



The developer misinterpreted the requirement for the algorithm.

**Question 4**

**1 / 1 pts**

Mr. Test has been testing software applications on mobile devices for a period of 5 years. He has a wealth of experience in testing mobile applications and achieves better results in a shorter time than others. Over several months Mr. Test did not modify the existing automated test cases and did not create any new test cases. This leads to fewer and fewer defects being found by executing the tests. What principle of testing did Mr. Test not observe?



Testing depends on the environment.



Defects cluster together.

**Correct!**



Repeating of tests will not find new defects.



Exhaustive testing is not possible.

**Question 5**

**1 / 1 pts**

In what way can testing be part of Quality Assurance?

**Correct!**



It contributes to the achievement of quality in a variety of ways



It ensures that standards in the organization are followed.



It measures the quality of software in terms of number of executed test cases



It ensures that requirements are detailed enough.

**Question 6**

**0.5 / 1 pts**

Match the test work product to its definition.

**Correct!**

**Test suite**



**You Answered**

**Test case**

Contains expected results.

**Correct!**

**Test script**



**You Answered**

**Test charter**

 An instruction of test goals and possible test ideas on how to test.

**Question 7**

**0 / 1 pts**

Which of the following is a typical test objective?

**Correct Answer**



Preventing defects

**You Answered**



Comparing actual results to expected results



Repairing defects



Analyzing the cause of failure

**Question 8**

**Original Score: 0 / 1 pts Regraded Score: 1 / 1 pts**

**This question has been regraded.**

A product owner says that your role as a tester on an Agile team is to catch all the bugs before the end  
of each iteration. Which of the following is a testing principle that could be used to respond to this  
statement?



Defect clustering

**Correct Answer**



Testing shows the presence of defects

**You Answered**



Absence of error fallacy



Root cause analysis

**Question 9**

**1 / 1 pts**

A tester participated in a discussion about proposed database structure. The tester identified a potential performance problem related to certain common user searches. This possible problem was explained to the development team. Which of the following is a testing contribution to success that BEST matches this situation?



Enabling required tests to be identified at an early stage



Ensuring processes are carried out properly



Reducing the risk of untestable functionality

**Correct!**



Reducing the risk of fundamental design defects

**Question 10**

**1 / 1 pts**

As a result of risk analysis, more testing is being directed to those areas of the system under test where initial testing found more defects than average.

Which of the following testing principles is being applied?



Testing is context dependent



Beware of the pesticide paradox



Absence of errors is a fallacy

**Correct!**



Defects cluster together

**Question 1**

**3 / 3 pts**

Briefly enumerate and describe the phases of the RIPR model.

Your Answer:

1. Reachability: Location(s) of where the error is in the program being tested.
2. Infection: The program being tested must be in an incorrect state.
3. Propagation: The program must be able to produce an undesirable result.
4. Reveal: The undesirable result must be witnessed by a tester.

See Life Cycle Testing lecture slides.

**Question 2**

**2 / 2 pts**

Associate the development phases with their corresponding testing levels.

**User requirements analysis**



**System requirements analysis**



**Global design**



**Detailed design**



**Implementation**



**Question 3**

**1 / 1 pts**

Code inspection is a type of:



Dynamic testing.



Static testing.



Model-based testing.



All of these.



Black-box testing.

**Question 4**

**1 / 1 pts**

The term "test basis" can refer to:



UML diagrams.



User manual.



Written requirements.



All of these.



Program source.

**Question 5**

**1 / 1 pts**

"You can find and fix a large number of defects but still have a system that doesn't meet user requirements" best exemplifies which principle?



Pesticide paradox



Absence-of-errors is a fallacy



Exhaustive testing is impossible



Defects cluster together



All of these

**Question 6**

**1 / 1 pts**

Test-Driven Development (TDD) is most closely associated with which testing level?



Regression testing



Integration testing



Component testing



Structural testing

**Question 7**

**1 / 1 pts**

Regression testing focuses on



Verifying the correctness of newly-introduced code



Lowering risk by comparing test results to requirements



Ensuring that previously tested software performs properly after a change



Validating that functionality properly regresses

**Question 1**

**1 / 1 pts**

The process of \_\_\_\_\_\_\_\_\_\_\_ is the process of testing code that has been modified.

**Correct!**



Regression testing



Re-testing



Test observation



Automated testing

**Question 2**

**1 / 1 pts**

Which of the following is not a benefit of test automation?



Reduces human error

**Correct!**



Reduces total number of test cases



Reduces variance in test quality between individuals



Reduces cost

**Question 3**

**1 / 1 pts**

The term "software testability" can be summed up as "how hard is it to find faults in the software?"

**Correct!**



True



False

**Question 4**

**1 / 1 pts**

The two main practical problems that determine testability include

**Correct!**



Observability



Repeatability

**Correct!**



Controllability



Recoverability

**Question 5**

**1 / 1 pts**

In the Arrange, Act, Assert model, the input values for a test case are established in the \_\_\_\_ step.

**Correct!**



Arrange



Assert



Act

**Question 6**

**1 / 1 pts**

In the Arrange, Act, Assert model, the actual versus expected results are normally tested in the \_\_\_\_ step.



Arrange



Act

**Correct!**



Assert

**Question 7**

**1 / 1 pts**

Which of these method name formulations is preferred in this class?

**Correct!**



<method name>\_Should\_<expected behavior>\_When<state under test>



Test<state under test>



<methodname>\_<state under test>\_<expected behavior>



<method name>Test<unique digit>

**Question 8**

**3 / 3 pts**

Why does data abstraction complicate testability?

Your Answer:

Data abstraction reduces controllability and observability. Therefore, it can become harder to see where defects can or will occur. Abstracted methods can also be harder to control.

**Question 1**

**1 / 1 pts**

In agile methods, when is software is judged "correct"?



When it behaves correctly on a specific set of tests



When it runs to completion on all possible combinations of inputs



Never. It is impossible to demonstrate that software is correct.



When it passes tests that exercise every code branch

**Question 2**

**3 / 3 pts**

What does "YAGNI" stand for?  In one or two sentences, explain what that means in terms of test-driven development.

Your Answer:

"You Aren't Going to Need It"

Write new code as it's needed and avoid speculating on future code that may be needed. If a functionality does not require more code, do not write it.

**Question 3**

**1 / 1 pts**

Which of the following is true of continuous integration?



Continuous integration points out regressions.



Mistakes caused by a single developer can be quickly identified.



All of these are correct.



Continuous integration includes monitoring of static analysis.



Continuous integration highlights divergent design decisions.

**Question 4**

**4 / 4 pts**

In a few short sentences, explain the "red, green, refactor" cycle in TDD.

Your Answer:

Start with a test method meant to fail or turn red when ran (because the tested method isn't implemented yet).

Right the method intended to be tested so that it passes or turns green.

Once the method works, make it pretty! AKA, refactor the new method.

"Write", not "Right."

**Question 5**

**1 / 1 pts**

Test sets with interdependent test suites are more robust, easier to maintain, and more likely to identify errors.



True



False

**Question 1**

**6 / 6 pts**

Match the terms on the left to their definitions on the right.

**Defect**



**Error**



**Failure**



**Testing**



**Debugging**



**Question 2**

**3 / 3 pts**

Explain the differences between black-box and white-box testing.

Your Answer:

White box testing is testing focuses on internal structures and the tester has access to the inner workings or code of the testing subject.

Black box testing is focused on the functionality of it's subject and the tester generally only has a surface level understanding of what they are testing.

**Question 3**

**5 / 5 pts**

Considering the "V model," match the software development activity on the left to the testing activity on the right.

**User Requirements**



**System Requirements**



**Global Design**



**Detailed Design**



**Implementation**



**Question 4**

**2 / 2 pts**

Which of the following are not among ISTQB's seven testing principles?



Exhaustive testing is impossible



Defects cluster together



Testing proves absence of defects



Beware of the poltergeist paradox



Absence of errors is a fallacy

**Question 5**

**1 / 1 pts**

Which of the following is not a unit testing framework?



NUnit



MsTest



xUnit



jUnit



FluentAssertions

**Question 6**

**5 / 5 pts**

Explain the "Red, Green, Refactor" cycle in Test-Driven Development.

Your Answer:

Start with a test method meant to fail or turn red when ran (because the tested method isn't implemented yet).

Write the method intended to be tested so that it passes or turns green.

Once the method works, make it pretty! AKA, refactor the new method.

**Question 7**

**3 / 3 pts**

Describe the YAGNI principle.

Your Answer:

"You Aren't Going to Need It"

Write new code as it's needed and avoid speculating on future code that may be needed. If a functionality does not require more code, do not write it.

**Question 8**

**2 / 2 pts**

**Use the class attached to the Exam introduction to answer this question.**

What is wrong with the code?

Your Answer:

First off, the code doesn't seem to be very useful. Because of the OR statement, the method only returns a count of positive numbers. Regardless of weather a positive number is odd or even, it is added to the count with this code.

I would change the negative test to check for positives since it is more concise code. Instead of checking for the % operator to return 1, it should check if it is 0 then NOT add it to the count. This would also count negative odd numbers assuming the business logic accepts negatives numbers as capable of being odd/even.

Odd or Positive implies all positive numbers but only negative odd ones. It might not be useful, but it does do something. Anyway, you solution is correct. Either "!=0" or "=-1" would work. That last comment is curious. What would make you consider the possibility that negative odd numbers shouldn't be counted, I wonder?

**Question 9**

**5 / 5 pts**

**Use the class attached to the Exam introduction to answer this question.**

Based on your answer to the previous question, correct the defect in the class and submit the .cs file with the correction.

***Upload ONLY the .cs file.***

[ArrayCheckClass.cs](https://bc.instructure.com/files/131375024/download)

**Question 10**

**0 / 3 pts**

**Use the class attached to the Exam introduction to answer this question.**

If possible, give a test case that does not execute the defect.

Your Answer:

As written, any odd number that int is not added to the final count. So a test case that does not contain any negative odd ints will not execute the defect. For example...

{ 1, 2, 3, 10 }

Expected result: 4

Incorrect. The array must be null or empty. All other inputs result in the fault being executed. This is very similar to some questions in the first two homework assignments.

**Question 11**

**0 / 3 pts**

**Use the class attached to the Exam introduction to answer this question.**

If possible, give a test case that executes the defect but does not result in a failure

Your Answer:

I do not believe this is possible. When the defect is executed, an unexpected and inaccurate result is returned. If the defect happens, it results in a failure. For example.

{ -1 , 2, -3, 5, 10 }

Expected result: 5

Actual Result: 3

In short, this defect always causes a failure.

No; the input you specified in Q10 executes the defect but doesn't result in failure, so it's not unavoidable.

**Question 12**

**13 / 12 pts**

**Use the class attached to the Exam introduction to answer this question.**

In a C# file:

1. Write a parameterized test case with at least three sets of values.  Ensure that at least one set of values demonstrates that the error has been corrected.
2. Write a test case that ensures that the NullReferenceException is properly thrown when searchArray is null.

***Upload ONLY the .cs file from your Unit Test project.***

For full credit:

1. Your code must execute properly against the class attached to the exam introduction.
2. You must follow the structural and naming conventions for the tests.
3. You must not use the ExpectedException attribute.
4. You must use the SUT alias.

For up to 3 points of extra credit: specify all assertions using FluentAssertions.

[OddOrPosShould.cs](https://bc.instructure.com/files/131377149/download)

-1 for failing to use the course naming standard for test methods . Don't include the name of the method under test in the test method name (i.e., test class should be OddOrPos\_Should. The first test method should be ReturnCorrectValue\_WhenInputCorrect. +2 for fluent assertions on one test

**Question 1**

**1 / 1 pts**

Use of decision tables is an example of a static technique.



True



False

**Question 2**

**2 / 2 pts**

Which of these are examples of static testing techniques?



Informal reviews



Code coverage analysis



Control Flow analysis



Walkthroughs



Path analysis

**Question 3**

**2 / 2 pts**

Which are true of specification-based techniques?



They are dynamic



They generally run in the compiler.



They are used in black-box testing



They are static

**IncorrectQuestion 4**

**0 / 2 pts**

In robust boundary value analysis using nominal values, what is the minimum set of test values for each valid range?



2



3



4



6



7



8

**Question 5**

**3 / 3 pts**

Describe equivalence class partitioning in 50 words or less.

Your Answer:

Equivalence class partitioning is taking a range in a class or software and spit values into categories based on their behavior. The values should be partitioned together by identical results and sequentially and be book ended by the ranges minimum -1 and maximum +1 values. So a range of 1 through 10 might look like...  
0 = invalid

1 - 5 = foo

6 - 10 = bar

11 = invalid