THIRD QUIZ: CS4472A Tuesday, 5 December 2017, 7:10 pm, Room MC17

NAME AS APPEARS ON STUDENT ID:

STUDENT ID NUMBER:

UWO/CONFLUENCE USER NAME:

REMINDERS:

1. (from course outline) The quiz will be closed book, closed notes, with no electronic devices allowed, with particular reference to any electronic devices that are capable of communication and/or storing information.
2. Write neatly. If the marker can't read it, it is wrong.
3. This exam shouldn't take long to write. On the other hand, time will pass. It is a 30 minute quiz with 20 questions. If you complete a question every minute you will still have 10 minutes at the end to double check that everything is in order.
4. While you are not allowed to open the exam booklet until the proctor says you can, you can fill out the information on the cover page. You should also get out your student id and make sure your pencils and pens are in order. If you need to get something out of your jacket or knapsack once the exam has started, raise your hand and wait til a proctor comes to you to oversee the matter.

1. To illustrate the relation between testing and software design, we will look at the programming technique ANSWER
   - test driven development
2. An important concept we will look at related to the question of when has one done enough testing is ANSWER
   - coverage
   - mutation
3. It is easy to make up test inputs, but it can be tricky to know what the right output for a given input should be. This is refered to as the ANSWER problem
   - Oracle
4. Testing is generally about finding errors that have already been made. This course also covers the topic of ANSWER, which is about trying to prevent errors from being made in the first place.
   - quality assurance
5. The first testing framework for Ruby that we are looking at is called ANSWER
   - minitest
6. The first tool for checking code quality for programs written in Ruby is ANSWER, which is described as a code smell detector.
   - reek
7. The number of quizzes CS4472 will have this semester is ANSWER
   - 3
8. The number of weekly practices that CS4472 will have this semester is ANSWER
   - 10
9. The number of practice reviews that CS4472 will have this semester is ANSWER
   - 4
10. The per cent of the total mark allocated for all the quizzes is ANSWER
    - 21
11. The per cent of the total mark allocated for all the weekly practices is ANSWER
    - 30
12. The per cent of the total mark allocated for all the practice reviews is ANSWER
    - 49
13. The practice technique advocated in this class is a modification of the ANSWER
    - Personal Software Process
14. A main theme behind the practice technique advocated in this class is that in order to improve your programming, ANSWER
    - you need data about your past programming

15. A common piece of information for people interested in programmer productivity to track is ANSWER

    - time spent
    - number of lines of code written
    - number of defects found

16. Although we often think of programs as taking inputs and producing outputs, a higher level view of what is going on is to think of the programs as ANSWER about how to take inputs and produce outputs.

    - encode knowledge

17. The scripts that were designed to aid the practice process assume that you will be uploading a copy of your work to BitBucket every time you ANSWER

    - record a note about your practice progress

18. The protocols for practice expect that the longest amount of time that you will practice before recording a note is ANSWER

    - 30 minutes

19. The total amount of practice time you can get credit for during a practice week is ANSWER

    - 3 hours

20. The four phases of testing (according to Whittaker) are: 1) modeling the software environment, 2) selecting test cases, 3) running and checking test cases, and 4) ANSWER

    - checking how well the testing is going

21. The testing technique called boundary value partition starts with the notion of breaking the space of inputs into ANSWER

    - regions of interest

22. Structural testing is another name for ANSWER

    - code-based testing
    - white-box testing

23. The kind of testing we do to make sure that when we change a program we do not break something that used to work is called ANSWER

    - regression testing

24. Using combinatorial testing, if I have 10 binary inputs, I only need to use ANSWER test cases (each a setting of each of the 10 inputs) to expect to find 98 per cent of the errors in the program.

    - 13

25. The ANSWER is a method developed by Watt S. Humphrey to help individuals improve their programming skills based on existing methods that had been developed to help organizations improve their product development capabilities.

    - Personal Software Process

26. The paper Orthogonal defect classification-a concept for in-process measurements was an example of people at IBM analyzing records of defects in order to ANSWER

- improve their process

27. When multiple methods of a class have the same parameters, this is a code smell called ANSWER
    - data clumping
28. When multiple methods of a class have the same parameters, that generally indicates that those parameters should ANSWER
    - be put into a class of their own
29. MicroTest (MiniTest subset) discourages the writing of tests that depend on side-effects of the previous test by ANSWER
    - running tests in random order
30. MicroTest (MiniTest subset)'s usage pattern is for the test class to inherit from Test so that Class.inherited can be used to ANSWER
    - get a list of test classes
31. MicroTest (MiniTest subset) uses public_instance_methods to ANSWER
    - find methods that begin test_
32. The TDD Cycle is ANSWER
    - Red Green Refactor
33. RSpec specifications are sometimes called ANSWER documentation
    - executable
34. Unlike MiniTest which is implemented as a class library, RSpec is implemented in Ruby as an ANSWER
    - domain-specific language
    - DSL
35. RSpec and Cucumber are tools designed to support the ANSWER style of software development
    - Behavior-driven development
    - BDD
36. The differences between RSpec and Cucumber result from the intent that Cucumber test files are meant to be readable by ANSWER
    - the customer and the programmer
37. The differences between RSpec and Cucumber result from the intent that RSpec test files are meant to be readable by ANSWER
    - just the programmer
38. The Capability Maturity Model for US government contractors distinguishes 5 levels of company software development process. Level 1 is characterized as ANSWER
    - chaotic
    - ad hoc
39. The Capability Maturity Model for US government contractors distinguishes 5 levels of company software development process. Level 5 is characterized as ANSWER

- continually improving

40. Many of the ideas of the Capability Maturity Model were adapted to individual developers under the name ANSWER

    - Personal Software Process

41. In the Testing Maturity Model, at Level 5, we aim at ANSWER rather than defect detection

    - defect prevention

42. The corporate policy of developers merging their working copies into the main line of the branch repository several times a day is called ANSWER

    - continuous integration

43. The motivation behind multiple merges per day per developer is to ANSWER

    - minimize merge conflicts

44. The pattern where you create an object whose job is to create other objects (rather than using new to create other objects) is called ANSWER

    - the factory pattern

45. While the notation looks odd, in RSpec, "it" is actually implemented in Ruby as an ANSWER

    - method

46. In MiniTest, we write test classes that inherit from Test, but in RSpec these test classes are actually being created at runtime by ANSWER

    - describe

47. Once RSpec has created a test class, it fills in its definition by executing the Ruby method ANSWER

    - module_exec

48. When I say that in RSpec, expect x.to eq y, eq an object that inherits from ANSWER, meeting the requirements of to

    - Matcher

49. Modified condition/decision coverage is often a requirement (regulatory or contractual) in ANSWER

    - safety-critical applications
    - avonic systems
    - automotive systems

50. The four requirements of MC/DC are: 1) each entry and exit point is invoked, 2) each decision takes every possible outcome, 3) each condition in a decision takes every possible outcome, and 4) ANSWER

    - each condition in a decision is shown to independently affect the outcome of a decision

51. One study of 198 user major failure reports on 5 widely used distributed systems found statement coverage testing could have caught nearly ANSWER of the causes.

    - a quarter
    - 25 per cent
    - 23 per cent

52. One study of 198 user major failure reports on 5 widely used distributed systems found that nearly all failures were caused by coding mistakes in ANSWER

    - the error handling code

53. One study of 100 large open source Java programs compared better code coverage with number of post-release defect reports and found ANSWER

    - no connection

54. A study by Ahmed et al found that the probability of errors in untested code was ANSWER the probability of errors in tested code

    - twice

55. The S in Solid stands for ANSWER

    - single responsibility principle

56. The O in SOLID stands for ANSWER

    - open/closed principle

57. The L in SOLID stands for ANSWER

    - Liskov substition principle

58. The I in SOLID stands for ANSWER

    - interface segregation principle

59. The D in SOLID stands for ANSWER

    - dependency inversion principle

60. According to Robert Martin who first promoted the SOLID methodology, the S doesn't refer to functions, but to ANSWER

    - roles in the business that uses the software

61. According to Michael Feathers, code that is difficult to test is ANSWER

    - poorly designed

62. Sandi Metz approaches unit testing of an object by first distinquishing among three different message origins: incoming, outgoing, and ANSWER

    - sent to self

63. Sandi Metz's approach to unit testing of objects also distinguishes two types of messages: command and ANSWER

    - query

64. Sandi Metz's approach to unit testing only does three kinds of tests on three of the six origin-type message categories. She says you should "assert the result" on messages of the origin-type ANSWER

    - incoming query

65. Sandi Metz's approach to unit testing only does three kinds of tests on three of the six origin-type message categories. She says you should "assert the direct public side-effects" of messages of the origin-type ANSWER

- incoming command

66. Sandi Metz's approach to unit testing only does three kinds of tests on three of the six origin-type message categories. She says you should "expect to send" messages of the origin-type ANSWER

    - outgoing command

67. The logs of the practice tasks are supposed to provide a basis for improving your process. A major way to operationalize such improvement is to generate a checklist of things to pay more attention to in the future. We saw an example of how to build such checklists when looking at how software companies improve their ANSWER process

    - code review

68. When considering mocking for tests, we should consider the reason for the test itself. For example, if we are doing acceptance tests for a customer, then mocking is ANSWER

    - not appropriate

69. When considering mocking for tests, we should consider the reason for the test itself. For example, if we are doing tests to prevent a bug from returning to the code base, then the amount of reality we need in the test is ANSWER

    - only enough to reproduce the bug

70. When considering mocking for tests, we should consider the reason for the test itself. For example, if we are doing acceptance tests for a customer, then mocking is ANSWER

    - not appropriate

71. If we are mocking a third party API, this can cause the problem that ANSWER

    - the third party API might change and the change not be reflected in the tests

72. When testing, it is important to realize that the program that is being tested is not some random piece of code generated by a room of monkeys with keyboards, but rather than it is probably pretty close to being correct. This assumption is called ANSWER

    - the competant programmer hypothesis

73. An important assumption in testing is that although any particular program failure may result from a complex sequence of events, there is generally a first event that if it hadn't gone wrong the failure wouldn't have occurred. Thus it is sufficient to test for these simple event failures rather than having to build tests targetted at complicated interaction failures. This effect is called ANSWER

    - the coupling effect

74. We know that if a line of code hasn't been executed by at least one of our tests, there are major gaps in our testing. However, missing lines of code are not very common errors for programmers to actually make. Much more likely is for a programmer to make a typo. Checking a test suite to see if it can find all typos is the idea behind ANSWER testing

    - mutation

75. Defect causal analysis involves the following steps: select problem sample, classify selected problems, identify systematic errors, ANSWER, develop action proposals, and document meeting results

    - determine principal cause

76. In defect causal analysis, the information you are trying to extract from the problem reports in order to classify them are: when the defect that caused the problem was inserted into the software; when the problem was detected; and ANSWER

    - what type of mistake was made
    - what type of defect was introduced

77. In the talk Cucumbers Have Layers, A Love Story, it was advocated that one do BDD testing with both cucumber and rspec on the same project. The role of cucumber would be to provide the customer ANSWER tests and the role of rspec would be to provide the unit tests

    - acceptance

78. In The Principles Behind the Agile Manifesto, the overriding theme is to satisfy the customer through ANSWER

    - early and continuous delivery of valuable software

79. Any agile team must refine and reflect as it goes along, ANSWER in its local circumstances

    - constantly improving its practices

80. The version of Agile that we looked at in class and that lead to the development of cucumber envisions interacting with the customer in a structured way through the discussion of ANSWER

    - user stories

81. In our Agile approach to determining what tasks we are to do for the customer next, the high level information that we want from the customer is what is the anticipated role in the organization of the person who would do the task, what is it that they want to do, and ANSWER

    - what benefit they expect to get from the task
    - what reward they expect to get from the task

82. In the discussion of extreme programming, one core practice that was presented was ANSWER. Here the idea is that projects with few delivery points often produce last minute "death marches" that produce poor code just to meet the minimal requirements to claim that made the target. Extreme programming wants to avoid this sort of last minute panic coding.

    - sustainable pace

83. In the discussion of extreme programming, one core practiceof interest was ANSWER. It was claimed that it produced better code quicker, most programmers were happy with it, and it help distribute project information across the team.

    - pair programming.

84. In the discussion of extreme programming, one core practice of interest was ANSWER. By this they meant that one should establish overriding themes for projects that then drive common systems of names making it easier to find things in the code.

    - metaphor

85. When developing tasks with the customer, one approach to what good tasks for the team to undertake look like is covered by the INVEST acronym. The I stands for independent. This means that a task shouldn't depend on ANSWER

    - other tasks

86. When developing tasks with the customer, one approach to what good tasks for the team to undertake look like is covered by the INVEST acronym. The S stands for small. Before cucumber was developed, this often meant that such a task was written on ANSWER to make them quick to read and easy to tack up on a scheduling board.

    - an index card

87. In Confident Code, Avi Grimm presented techniques for refactoring code to improve its readability, i.e., narrative structure. One approach is to replace usages of nil with a special ANSWER that responds to messages by just returning itself. This means method invocations can be pipelined with constant checks that the previous stage didn't produce a nil.

    - Null Object

88. The Confident Code talk discussed many ways to avoid using nil in code – which tends to require other pieces of code to worry about whether they have recieved a nil. In addition to returning an object that returns itself in response to any message, the other major approach is to ANSWER

    - raise an error

89. In Confident Code, there was a general advocacy of minimizing the use of if. It was claimed that this helps testing because ANSWER.

    - there are fewer paths that have to be covered
    - there are fewer paths to test

90. Many people have noticed that Ruby doesn't have static types like Java. However, it is possible to insert type definitions into Ruby code using the DSL ANSWER

    - contracts

91. When using the Ruby DSL for dynamic types, one can use builtin types like Num, but one can also define one's own types, like Even that requires a value of that type to be an even number. To do this, we create a class Even that contains the method ANSWER, which checks to see if its input parameteris even.

    - valid?

92. Amanda Laucher gave an interesting talk on Types vs Tests. Her view was that whether you used a type or a test to check some property of a program depended on the kind of property involved. If the property was that something specific exists, then you would ensure this with ANSWER

    - a test

93. Amanda Laucher gave an interesting talk on Types vs Tests. Her view was that whether you used a type or a test to check some property of a program depended on the kind of property involved. If the property was that something always had a certain quality, then you would ensure this with ANSWER

    - a type

94. Amanda Laucher gave an interesting talk on Types vs Tests. She seemed to indicate that one of the reasons we do so much testing is because the type systems in the languages we use (such as Java) are not expressive enough. Although she mentioned many languages that were experimental research sort of languages; she mentioned ANSWER as the common language for people who are really into types.

    - Haskell

95. The most primitive web testing system we talked about in class was ANSWER

- selenium
- selenium webdriver

96. The main high level web testing notation that cucumber and rspec use is ANSWER
    - capybara