SECOND QUIZ: CS4472A Tuesday, 7 November 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. The first tool for checking code quality for programs written in Ruby is ANSWER, which is described as a code smell detector.
   ANSWER=

2. The D in SOLID stands for ANSWER
   ANSWER=

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

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

5. MicroTest (MiniTest subset) discourages the writing of tests that depend on side-effects of the previous test by ANSWER
   ANSWER=

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

7. MicroTest (MiniTest subset) uses public_instance_methods to ANSWER
   ANSWER=

8. The I in SOLID stands for ANSWER
   ANSWER=

9. When multiple methods of a class have the same parameters, this is a code smell called ANSWER
   ANSWER=

10. 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
    ANSWER=

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

12. To illustrate the relation between testing and software design, we will look at the programming technique ANSWER
    ANSWER=

13. 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
    ANSWER=

14. 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
    ANSWER=

15. RSpec and Cucumber are tools designed to support the ANSWER style of software development
    ANSWER=

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

17. The per cent of the total mark allocated for all the quizzes is ANSWER
    ANSWER=

18. Structural testing is another name for ANSWER
    ANSWER=

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

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

```
exam_database_file= examdatabase.json
exam_format= latex
dump_database= false
line_width= 72
question_count= 20
create_exam= false
answer_key= true
sample_seed= 41903
shuffle_seed= 999
["C1", "C2", "C3", "C4", "C5", "C6"]
["C1", "C2", "C3", "C4", "C5", "C6"]
```

1. reek
2. dependency inversion principle
3. method
4. regions of interest
5. running tests in random order
6. 30 minutes
7. find methods that begin test_
8. interface segregation principle
9. data clumping
10. Oracle
11. minimize merge conflicts
12. test driven development
13. the factory pattern
14. record a note about your practice progress
15. - Behavior-driven development
    - BDD
16. module_exec
17. 21
18. - code-based testing
    - white-box testing
19. continuous integration
20. defect prevention