FIRST QUIZ: CS4472A Tuesday, 3 October 2017, 7:10pm, 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 Capability Maturity Model for US government contractors distinguishes 5 levels of company software development process. Level 5 is characterized as ANSWER
   ANSWER=
2. The D in SOLID stands for ANSWER
   ANSWER=
3. 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=
4. 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.
   ANSWER=
5. The per cent of the total mark allocated for all the quizzes is ANSWER
   ANSWER=
6. 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.
   ANSWER=
7. The number of practice reviews that CS4472 will have this semester is ANSWER
   ANSWER=
8. The first testing framework for Ruby that we are looking at is called ANSWER
   ANSWER=
9. When multiple methods of a class have the same parameters, this is a code smell called ANSWER
   ANSWER=
10. The differences between RSpec and Cucumber result from the intent that Cucumber test files are meant to be readable by ANSWER
    ANSWER=
11. Many of the ideas of the Capability Maturity Model were adapted to individual developers under the name ANSWER
    ANSWER=
12. MicroTest (MiniTest subset) discourages the writing of tests that depend on side-effects of the previous test by ANSWER
    ANSWER=
13. The protocols for practice expect that the longest amount of time that you will practice before recording a note is ANSWER
    ANSWER=
14. The testing technique called boundary value partition starts with the notion of breaking the space of inputs into ANSWER
    ANSWER=
15. 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=
16. The motivation behind multiple merges per day per developer is to ANSWER
    ANSWER=

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

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

19. The first tool for checking code quality for programs written in Ruby is ANSWER, which is described as a code smell detector.
ANSWER=

20. 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
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= 2322
shuffle_seed= 245
["C1", "C2", "C3", "C4", "C5", "C6"]
["C1", "C2", "C3", "C4", "C5", "C6"]
```

1. continually improving
2. dependency inversion principle
3. record a note about your practice progress
4. encode knowledge
5. 21
6. 13
7. 4
8. minitest
9. data clumping
10. the customer and the programmer
11. Personal Software Process
12. running tests in random order
13. 30 minutes
14. regions of interest
15. continuous integration
16. minimize merge conflicts
17. describe
18. twice
19. reek
20. regression testing