MONASH
University

EXAM CODES: FIT2107

TITLE OF PAPER: Software Quality and Testing Mock Exam

EXAM DURATION

Rules

During an exam, you must not have in your possession any item/material that has not been authorised for your exam. This includes books, notes, paper, electronic device/s, mobile phone, smart watch/device, calculator, pencil case, or writing on any part of your body. Any authorised items are listed below. Items/materials on your desk, chair, in your clothing or otherwise on your person will be deemed to be in your possession.

You must not retain, copy, memorise or note down any exam content for personal use or to share with any other person by any means following your exam.

You must comply with any instructions given to you by an exam supervisor.

As a student, and under Monash University's Student Academic Integrity procedure, you must undertake your in-semester tasks, and end-of-semester tasks, including exams, with honesty and integrity. In exams, you must not allow anyone else to do work for you and you must not do any work for others. You must not contact, or attempt to contact, another person in an attempt to gain unfair advantage during your exam session. Assessors may take reasonable steps to check that your work displays the expected standards of academic integrity.

Failure to comply with the above instructions, or attempting to cheat or cheating in an exam may constitute a breach of instructions under regulation 23 of the Monash University (Academic Board) Regulations or may constitute an act of academic misconduct under Part 7 of the Monash University (Council) Regulations.

Authorised Materials

OPEN BOOK	☐ YES	☑ NO
CALCULATORS	☐ YES	☑ NO
DICTIONARIES	☐ YES	☑ NO
NOTES	☐ YES	☑ NO
SPECIFICALLY PERMITTED ITEMS	☐ YES	☑ NO

if yes, items permitted are:

Instructions

This mock exam is of a similar format and the questions are indicative of some of the types of questions you might see on the real exam. Some of them are directly from past exams in FIT2107; some have been modified, and some are completely new. I think, if anything, they are slightly harder than the real exam will be.

There is a total of 54 marks available in this exam. The number of marks allocated to a question is a rough guide to how long you should spend on it.

QUESTION 1: (12 Marks)

Programmable Logic Controllers (PLCs) are a specialized type of embedded computer system used by engineers to control industrial processes. They can be used to control motors, heating elements, valves (automated taps to control liquid or gas flow), and many other devices that are present in automated factories. They respond to a variety of sensors, including "on/off" sensors as well as analogue sensors that measure things like temperature, pressure, and speed.

Rather than conventional programming languages, PLCs are programmed using a specialized graphical language – the "programs" look like electrical circuit diagrams. They can often specify exact timing constraints – for instance, that a motor is to stop 50 milliseconds after a temperature sensor reads a certain value. Programs for a PLC are written on PCs and uploaded to PLCs via a USB or network connection and interpreted by the PLC firmware.

The factories, and the hardware that these PLCs are used to control, often lasts for decades.

You are responsible for developing the firmware for a new PLC model. The firmware acts as the operating system for the PLC – it provides the environment for the programs written by the engineers for the specific factory installation to run. It includes the interpreter for the PLC programming language, and facilities to upload new or modified programs. It does not include the development environment that runs on the PCs, nor the programs written for a specific factory installation.

System quality attributes are defined in ISO/IEC9126 as:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

For each quality attribute, give two examples of a specific, assessable requirement/property that might plausibly be applicable for the PLC firmware. If an attribute is of limited or no relevance in this application and therefore you can't give two such examples, explain why. If the attribute is relevant to the application, but it is difficult to describe assessable requirements relating to it, explain why.

Your understanding of the system quality attributes is much more important than your knowledge of the specifics of PLCs, so your answer does not have to go into great detail about the operational aspects of factories or other systems that PLCs are used to control.

1 mark per reasonable requirement/property.

2 marks if they provide a reasonable explanation of why property is of limited/no relevance. Partial marks granted if explanation is not completely insane but somewhat unclear or slightly off-topic.

Some example answers (note – you do NOT have to give these exact answers to receive the marks, these are purely to give an indication of acceptable answers to this question might be):

Functionality:

Administrator must be able to upload a new program to the PLC via Ethernet connection.

System must be able to switch a gas load valve on and off according to an uploaded program.

Reliability:

PLC must be able to execute an uploaded program continuously for 1 week without interpretation errors.

PLC must restart execution of uploaded program after a power failure.

Usability:

As this device has such a limited user interface, usability is of limited relevance here.

Efficiency:

Programmed actions should take place within 1ms of the specified response timeframe.

The system must be able to execute programs containing 1000 diagram elements

Maintainability:

Very hard to give quantifiable maintainability requirements in general, so would accept that it's not easy to give testable maintainability requirements for this system.

Would accept, for instance: firmware must be updateable via USB stick or network.

Also, would accept something related to level of unit test coverage, e.g. automated unit tests for the system must cover 90% of statements.

Portability

Would accept that it's of limited relevance given firmware is designed for specific hardware.

QUESTION 2: (8 + 4 + 4 + 2 + 6 + 4 = 28 Marks)

The questions relate to the following Python source code:

```python
1 class Student:
2    def __init__(self, firstName, familyName, grade):
3         self.givenName = firstName
4         self.familyName = familyName
5         self.grade = grade
6
7    def letter_grade(self):
8         '''Return the letter grade from the numerical mark
9         for this student according to the standard Monash scale'''
10        if self.grade >= 80:
11            return "HD"
12        elif self.grade >=70:
13            return "D"
14        elif self.grade >=60:
15            return "C"
16        elif self.grade >=50:
17            return "P"
18        else:
19            return "N"
20
21    def pass_subject(self):
22         '''Return true if the student has received a passing grade for
23         this subject'''
24         return self.letter_grade() is not "N"
25
26    def result_summary(self):
27          return self.firstName + " " + self.familyName + " " +
str(self.grade) + " " + self.letter_grade()
28
29 class Unit:
30    def __init__(self, students):
31          self.students = students
32
33    def fail_report(self):
34         for student in students:
35             if not student.pass_subject():
36                  print(student.result_summary())
```

a) Draw CFG for the letter grade method? (8 Marks)

   Note: It is a hybrid question so you can draw the CFG on the piece of paper, take picture and upload it as an answer. You can also use a tool to draw it. However, in the real exam draw the diagram on a piece of paper.

b) Give the smallest possible set of test inputs (that is, values for self.grade) that would achieve 100% statement coverage for the method letter_grade() (4 Marks)

   For example ([80, 70, 60, 50, 49]) would do.

   1 mark if statement coverage achieved, 1 mark for smallest.

c) Give the smallest possible set of test inputs for Unit.fail_report() (where a test input involves calling the constructor to create a Unit object and calling fail_report() on it) to achieve 100% branch coverage for fail_report() (not including methods invoked by it). (4 Marks)

I don't care how you express this, but I'd say something like,

Myunit = Unit([Student("Fred", "Bloggs", 48), Student("Jane", "Nguyen", 70)]). 1 test input, a list with two students in it!

2.5 marks if you have test inputs that achieve branch coverage, 1.5 marks if it's the smallest set (1 input)

d) Do you think it would be a good idea to test fail_report() just using your answer to part 4? Explain why or why not and if not, provide an alternative approach in about a paragraph. (2 Marks)

1 mark Testing for all the possible conditions is kinda bundled up in a single test, which makes it harder to debug than necessary.

1 mark Suggest you split the cases up into separate tests (eg an empty list).

e) Suggest an appropriate black-box testing strategy for letter_grade(). Explain why it is appropriate for this method, and use it to come up with a set of test cases for letter_grade() (6 Marks)

2 marks for a clear strategy
1 mark for explanation Generally, equivalence testing is highly suitable for this as the input is one number with well-defined partition boundaries.
2 marks for tests following that strategy, 1 mark if there are clear statements of expected outputs.

f) Write Python code using the unittest module that implements the first of your tests in part e). (4 marks)

4 marks: correct Python code that would execute the test (modulo minor syntax errors).
2 marks: something that involves defining a test and checking an assertion.

QUESTION 3: (3+ 3 + 4+ 4 = 14 Marks)

One of the Chidamber and Kemerer metrics is Coupling Between Objects. It is defined as:

> CBO for a class is a count of the number of other class to which it is coupled…ie methods of one use methods or instance variables of another. (Chidamber and Kemerer (1994))

a) In general, do you think a system with many classes with a high CBO metric would be easier or harder to unit test well than an equivalent system with lower CBO values? Explain your answer in about a paragraph. (3 marks)

It's going to be harder, in general, because the more classes that interact with each other, either the more bugs are going to be the result of interaction rather than within the class, the harder the results are going to be to interpret, or the more mocking you'll have to do.

1 mark for just stating "harder", 2 marks for quality of explanation (2 marks for a clearly argued paragraph that makes key points along the lines of that above, 1 for a partial or poorly structured justification).

b) Is it likely to be easy to reuse classes with very high CBO score? Justify your answer. (3 marks)

In general, no, because high CBO scores mean that it depends on a whole bunch of other classes, all of which have to be available and relevant to the reuse. Makes it hard to repurpose

1 mark for just stating "harder", 2 marks for quality of explanation (2 marks for a clearly argued paragraph that makes key points as above, 1 for a partial or poorly structured justification).

c) Are there aspects of good design related to coupling that CBO does not measure adequately? Explain your answer – if there are such aspects, include examples to illustrate. (4 marks)

Clearly, yes, as CBO does not take into account the nature of the coupling between the classes. There are different types of ways a class can be coupled, and not all of them are as tight. For instance, access to shared data, such as public attributes, is a much harder-to- change and bug prone way for classes to be coupled than access through a well-defined API.

1 mark for yes, 2 marks for quality of explanation, 1 mark for clearly stated example.

d) Chidamber and Kemerer's Response for a Class metric (RFC) is defined as the size of the set of the following methods:
- The methods defined in the class.
- Any method called directly by a method defined in the class.

If RFC is high, does that tell us anything about the likely value of the CBO score? What about the other way around – if CBO is high, is RFC likely to be high? Explain your answers. (4 marks)

If RFC is high, CBO will often be high, but not always. Could call many methods in one or a few classes. A single-class program would have a very high RFC but a CBO of 0.

If CBO is high, RFC will generally be high. Possible exceptions include if all coupling is via public attributes, but this would be very bad design and uncommon in real systems.

2 marks for each answer. For full marks, must clearly consider trends and possible exceptions, and explain clearly. 0.5 marks for a reasonable trend answer with no further explanation.