COMP33812 Examination Performance Feedback
2013–2014

Question 1

a) i) The part was looking for different sources of information about a system to be identified. Several answers described different elements of the documentation, but these are all just parts of the documentation source. Another popular source was to talk to the developers. However, the description of the scenario indicated that the company that developed the application no longer existed, from this it should have been possible to realise that access to the developers would not be feasible. Finally, the question indicated that the source code should not be one of the sources given in an answer. However, some answers explicitly said look at the course code, while others suggested looking at the structure of source code or the comments that it contained; both of these are part of the source code.

ii) Question was looking for how top-down reading gives good overview of system, bottom-up reading gives localised understanding and in practice tend to need a combination of both; top-down to find the program area of interest and bottom-up to understand this area in detail. Answers were general along these lines.

b) The part was looking for an explanation of how both code idioms and coding conventions help in software comprehension; i.e. that they give the programmer something that they are familiar with which means that understanding requires less effort. A significant number of answers defined what code idiom and coding conventions are without any attempt to explain how they help in code comprehension. Other answers assumed that code idioms and coding conventions were the same, which they are not. Finally a few answers totally ignored coding conventions.

c) Generally the control flow graph produced for this part was well done. The two major places where marks were lost was in not representing the for-loop as three nodes (initialise, condition and increment) and not including an exit arc. Marks were also lost for not labelling the arcs exiting condition nodes with true and false labels, and including nodes where there was no statement (in particular for the closing brace of code blocks).

d) This part was looking for the identification of the process each element of a collection idiom. As part of this, the question required a language independent form and identification of the line of code that equated to each part of this form. A significant number of answers lost marks by not giving the language independent form or by not identifying the line numbers for each part. A number of answers claimed to have identified a search within a collection idiom. In the code given, there is no search operation implemented; the code does call a method those name includes the phrase find.

Question 2

a) This part was looking for identification of two problems with Big Up front Design, e.g. requirements are never fully understood and requirements change. Several answers lost marks by only identifying one reason.

b) This part specifically asked for a description of evolutionary design at both the code and system levels and how these addressed problems with BUFD. So, for example, answers could mention code refactoring at the code level and early integration via slicing at the system level. A significant number of answers lost marks by giving generic descriptions of these techniques not and describing how/why they avoided problems with BUFD. Other answers lost marks by only addressing evolution at the code level and ignoring that at the system level. A number of answers discussed migration at the system level. This technique applies to the modernising of legacy systems and so does not truly address evolution during the development phase.

c) This part specifically asked for a description of what a code smell is and how it relates to evolutionary design and refactoring; i.e. that the presence of a code smell indicates a need to refactor. A number of answers gave detailed examples of code smells and failed to relate these to evolutionary design and refactoring.

d) This part was looking for the identification of three bad code smells in the code given; e.g. the use of meaningless names, the presence a of comment and two very similar methods. Most answers got the first
of these two smells, but did not identify the third. A few of the answers indicated a significant lack of understanding of Java classes and regex expressions. For example, some answers indicated that the regex expression on line 14 matching any sequence of characters was a file extension being appended. Whereas others indicated that the body of the toString method (lines 21-28) could use an immutable String class instead of the mutable StringBuffer class used. Perhaps the best one was that there were if statements without an else part, so the program would crashes if the condition was not met.

e) This part was a new style of question and consequently provided flexibility in how it could be answered. However, the question was clearly asking for characteristics and questions about the language. A number of answers attempted to describe, incorrectly, what the code was doing rather than the questions that should be asked to understand the language in which it was written. In future, this style of question is likely to ask for justifications for questions raised.

Question 3

a) This question was asking for evaluation of options from the consultant’s report and a recommendation to the board. A significant number of answers where generic and did not take into account the circumstances described in the scenario given. Some answers also lost marks by not giving a recommendation to the board.

b) This question was asking for evaluation of options on how to keep a shared database up-to-date. In other words, for each option (e.g. triggered push, timed push, timed pull) how current the data would be and the resources required to achieve it. A significant number of answers where generic and did not take into account the circumstances described in the scenario given.

c) This part was looking for an architecture that would support the operations of the combined companies. Almost all of the answers allowed some of the operations to be supported. However, a significant number of answers failed to link customers, or customer UI, with stock control. Thus, it would be impossible for a customer to know if an item was available.

d) This part was looking for details of the migration operation related to the scenario given. Many of the answers where generic and did not evaluate the order in which components would be migrated to the business value in the scenario given.

Question 4

a) This part was looking for three consequences of using open source components in relation to software evolution. Too many answers gave generic points that were not software evolution specific. A few answers also claimed that if the developers of the open source software were no longer active, it would not be possible to continue to use the software. However, as you still have the source of the open source code, this cannot be the case.

b) This part was looking for an in-depth discussion of the options available. A significant number of answers lacked real depth. Two major points were not considered in a significant number of answers. The first of these was the domain knowledge that would be required within the development team to implement the required algorithm; the scenario given would require a significant depth of domain knowledge for a successful implementation. The second point was licensing costs versus development costs; development costs are semi-predictable, licensing costs can be subject to political decisions and can vary widely.

c) This part was looking for thoughts about using open source software in particular how code changes over time. Answers consistently seemed to assume that the quality of the open source software would be constant. However, the scenario very clearly indicated a development phase for the open source software. Experience with all software development is that its quality or robustness improves over time. Thus, early versions of the open source software would probably not be as good as the commercial offering, however, eventually it would.