THE BCS PROFESSIONAL EXAMINATIONS
Professional Graduate Diploma

April 2007

EXAMINERS' REPORT

Programming Paradigms

General comments

Notwithstanding the steadily increasing number of candidates sitting this paper, it continues to be one on which they perform well, with a pass rate around 91%. Candidates seem to have been properly prepared for the examination and to understand the material, despite its advanced nature.
Questions 1, 2, and 3 were by far the most popular. Question 4 was answered badly but otherwise there was no significant difference in how well the candidates answered the questions.

Question 1

a) Language standardisation is often a lengthy and laborious process, taking up to 10 years to complete in some cases. Discuss why a standard for each language is needed and what might be the consequences of not having one. **(13 marks)**

[Syllabus section 7a, Nature of Programming Languages]
Standardisation is the process of producing a technical standard that can be used amongst both the competing entities producing the products and the users of the product, such as C++ or Java. It is vital everyone "sings from the same hymn sheet", otherwise it makes it impossible to use. For example, if there were no C standard, it would be necessary to learn different versions of C for different C compilers.
Other points to mention – standardisation helps:

- *to reduce inconsistencies between different implementations and thus make programs (and programmers!) more portable;*
- *to give all users an agreement of what is in a language;*
- *to establish world wide recognition of what should be in a language;*
- *to enable communication with different stakeholders;*
- *to facilitate interoperability between languages, e.g., Java's native interface (JNI).*

This is the first time that a question on language standardisation has been asked. Many candidates produced interesting answers regarding this area and achieved good marks. To gain high marks, candidates needed to answer the question as set; that meant discussing the consequences of not having a language standard. Some candidates did badly because they wrote about coding standards rather than language standards. Other candidates interpreted the question as asking for a comparison of different programming paradigms.

b) Discuss how the concepts of objects, classes and inheritance support the development of 'good' software. Illustrate your answer with either code or diagrams. **(12 marks)**

[Syllabus section 7c, Object Orientation]
Candidates were expected to discuss how these concepts help with designing and implementing a program. They should first of all have described the concept and then discussed how it helps to develop an application. For example, inheritance allows the design of a complex hierarchy not possible with non-OO languages, which arguably allows a more natural design.

Examples from an OO programming language such as C++, Smalltalk or Java should have been included to illustrate the concepts, or alternatively a diagram from an OO design method, such as UML.
Most candidates made a good attempt at this part, giving adequate descriptions of objects, classes and inheritance. Some candidates lost marks because they did not reflect on how these concepts could help software design. A few candidates produced pages of code to illustrate the concepts, but with little discussion of their usefulness. Examples were required, but a discussion of their value was also important to gain a higher mark.

Question 2

Structured programming changed the way programmers developed software systems in the 1970s and these changes were reflected in the style of the programming languages that were used. Since then there have been many other changes in the way in which programs are developed and these changes too have been reflected in new programming paradigms.

Choose two different programming paradigms and discuss how they reflect changes in the development of software, indicating any disadvantages that may be associated with the changes. **(25 marks)**

[Syllabus section 7a, Nature of Programming Languages]
Object-orientation, scripting and event-driven programming are examples of the new paradigms expected here, rather than functional or logic programming. A description of the characteristics of the chosen paradigms should have been given, along with an evaluation of their strengths and weaknesses.
Within their discussion candidates needed to discuss what are the key features of the paradigm that make it revolutionary. For example, object-orientation was a major paradigm shift from previous languages, introducing concepts such as inheritance, message passing, etc. that had previously been seen only in small scale, experimental languages. A basic answer would have covered the description of two paradigms, with some evaluation. To get a good mark, the discussion was expected to include an evaluation of what made the paradigm revolutionary and also to reflect on the drawbacks.
This question was on the whole well answered. Some candidates lost marks by just concentrating on one paradigm and then describing it in detail, e.g., explaining the key characteristics of object-orientation (typically, inheritance, encapsulation, classes and objects) and also provided a lot of code to illustrate the features. They then failed to say how the particular paradigm changed the development of software.
Many candidates failed to discuss the disadvantages of the two approaches they had chosen, either concentrating on the advantages, or just describing the features.

Question 3

[Syllabus section 7d, Functional Programming]

The selection of two different aspects of functional programming was intended to allow candidates to demonstrate their breadth of understanding of the issues involved.

a) Discuss the role of constructors, pattern matching and recursion, when using compound types within an applicative (functional) programming language. Provide illustrative examples of their role within a functional programming language. **(15 marks)**

Candidates were expected to describe how, for example, lists can be created using constructors which in turn can be used when defining functions by pattern matching. The importance of recursion in these functions should have been discussed, as in the absence of "statements", it is the only way of creating loops in expression evaluation. Candidates were required to provide illustrative examples of the respective roles that could describe, for example, how to define a (recursive) type for trees whose nodes *are labeled with integers, and in turn, how this type can be used to write functions that process the tree.*

In general the answers to this part of the question were very good and a few were excellent. There was a good use of examples to illustrate constructors, pattern matching and recursion, and these examples were based upon more than one functional programming language.

b) Distinguish between lazy and eager evaluation in the implementation of applicative (functional) programming languages. **(10 marks)**

Candidates were expected to define the terms used, and provide suitable illustrative examples for clarification. The importance of the implications (and hence advantages) of each evaluative mechanism should have been demonstrated in real terms, which may relate to storage, or time-based outcomes.

There were many good answers to this part, and some very good examples. The majority of responses focused upon the time-based issues associated with lazy and eager evaluation and only a few answers discussed the implications for storage requirements during evaluation.

Question 4

[Syllabus section 7e, Logic Programming].

The selection of two different aspects of logic programming was intended to allow candidates to demonstrate their breadth of understanding of the issues involved.

a) Discuss the suitability of pure logic programming for the development of interactive machine control systems such as an aircraft fly-by-wire system. **(12 marks)**

Candidates were expected to demonstrate their knowledge of the basic concepts of pure logic programming. Based upon their familiarity with a real logic programming language, candidates were expected to realise that in the case of the given example 'extra-logical' features are required, which are not in keeping with a pure logical approach. Candidates were expected to discuss issues such as input/output, program control and, real-time performance.
Whilst there were a number of reasonable answers, this part was answered poorly, with few candidates understanding the issues involved.

b) There are two common approaches to the concurrent execution of logic programming clauses, namely and-parallelism and or-parallelism. Compare and contrast these approaches. **(13 marks)**

Candidates were expected to provide a foundation to their responses by a brief discussion of the environment of concurrent execution. The logical background to such clause evaluation should have been given, and the two types distinguished. This would then have allowed them to be compared, using suitable examples to clarify the answer.
This part was answered very badly.

Question 5

To what extents are the issues of concurrency and distribution addressed by languages following the object oriented, functional and logic-based paradigms? Illustrate your answer with suitably annotated examples of each system, pointing out the relevant parts of the paradigm referred to in each instance. **(25 marks)**

[Syllabus section 7f, Related Issues.]

Candidates' responses were expected to address clearly the three paradigms referred to, which required them to identify the aspects of each paradigm relevant to the issues of concurrency and distribution. In turn, the illustrations were expected to be based on existing languages, although credit was given for any correct and relevant illustration . It was important that answers concentrated on the central issues and that these were not confused by discussion of peripheral matters such as performance.

It was expected that reasonable examples of each issue would be given for each paradigm, so that a 2 × 3 structuring of the response would be appropriate. Credit was given for following this pattern, as long as connected arguments are provided. Less structured answers were less favourably received.

There were a number of excellent answers to this question, although many candidates produced badly structured answers. Many answers provided suitably annotated examples and therefore attracted higher marks.