THE BCS PROFESSIONAL EXAMINATIONS
BCS Level 6 Professional Graduate Diploma in IT

April 2008

EXAMINERS' REPORT

Programming Paradigms

General Comments

The number of candidates sitting this paper fell by some 25% this year but the pass rate remained high at around 95%. Candidates seem to have been properly prepared for the examination and to understand the material, despite its advanced nature.

Questions 1 and 2 were, as always, by far the most popular. Question 3 was very unpopular and was comparatively badly answered.

Question 1

a) Describe the features an Interactive Development Environment (IDE) should have to support a team of software developers.

(10 marks)

[Syllabus section 7b, Programming Environments]
Candidates were expected to list a range of program development tools typically found in an IDE and to describe what they do. Candidates could have described, for example, debuggers, testing tools, linkers, configuration tools, loaders, screen painters, code generators, etc. A better answer would also have considered tools to support the team aspects of software development such as tools that allow modules to be checked-in and out or tools to support version managemen. Such tools are important if multiple programmers are working on the same area, particularly for teams not based physically close to each other.

b) Discuss the advantages and disadvantages of using such an IDE, evaluating how it aids productivity and improves the quality of the code produced.

(15 marks)

[Syllabus section 7b, Programming Environments]
In this part of the question, candidates were expected to reflect on the real usefulness of these tools. They should have indicated why they think the tools are important or not. For example, they could argue they improve the speed and performance of the programmer, help with debugging, testing, etc. Therefore programmer productivity is increased. Most things have disadvantages too and these could include that the programmer may fall into bad programming practices by relying on the tool to correct mistakes, so encouraging hacking, rather than getting the programmers to check and test their code thoroughly themselves.
For both parts, examples are expected of appropriate tools, for example, Microsoft Visual Studio, Borland C++, Borland JBuilder, etc. They needed to identify what elements of the tool are useful to productivity and say why they think are beneficial or not in a team environment.
This proved to be a popular question and on the whole was well answered. Part a) in particular often attracted high or full marks. Most candidates could describe the main

features of an IDE; marks were generally lost by not discussing how the features of an IDE could be applied to a team environment.

Part b) was also generally answered well. Candidates could normally describe the advantages of an IDE and went into these in great depth. For this part, marks were lost where candidates concentrated solely on the advantages, giving little thought to any disadvantages. Other answers concentrated only on describing the advantages and disadvantages, with little evaluation on how IDEs increase productivity and help improve the quality of the code produced. A higher mark went to candidates who did include this evaluation.
99% of candidates attempted this question, of whom 95% achieved a pass mark on it.

Question 2

A software company currently uses conventional (procedural) programming languages for its systems development. It is now considering moving to an object-oriented programming language. Discuss the effects of changing from a conventional to an object-oriented paradigm. Within your answer consider the benefits, overheads and disadvantages of this approach. Illustrate your answer with appropriate examples. In the programming world, there exists a variety of languages that a programmer can use to develop a software system.

(25 marks)

[Syllabus section 7c Object Orientation]
The candidates were expected to consider the advantages and disadvantages of using an object-oriented (OO) language such as C#, C++ or Java. The answer should have included a discussion of the concepts found in OO languages to aid the development of a system, for example, inheritance, polymorphism, encapsulation, information hiding.
Candidates needed to reflect on what benefits or problems these concepts bring to program development compared to traditional development techniques, e.g., reuse. Examples from an OO programming language should have been included to illustrate the points made.
This was a popular question that produced good answers overall. A lot of candidates focused on discussing the advantages of the object-oriented approach and went into great length in describing the features. Some discussion was needed of the disadvantages and overheads of this approach too. Concise examples of concepts were welcome and gained credit, but it was not necessary to write several pages of code with little explanation or description.
The candidates who scored the highest marks, were those who could reflect on the impact of changing from a conventional language to an object-oriented paradigm.
90% of candidates attempted this question, of whom 96% achieved a pass mark on it.

Question 3

a) Discuss the role of higher-order functions within the implementation of applicative (functional) programming languages.

(12 marks)

b) In a functional programming language, an expression is evaluated within the context of an environment. Discuss this statement, commenting upon any similarities, or otherwise, that might exist between functional and imperative programming languages.

(13 marks)

[Syllabus section 7d, Functional Programming]
The selection of two different aspects of functional programming was designed to allow candidates to demonstrate their breadth of understanding of the issues involved.
In part a) of the question, candidates were expected to define the nature of higher-order functions, and provide suitable illustrative examples of their role within a functional program. It was expected that the importance of such functions would be discussed, in particular the map function, which should have highlighted a major difference between functional and imperative languages. Whilst candidates introduced higher-order functions, mapping, sorting algorithm illustrations, and variable storage minimisation, the concept of functions having functions as arguments was not clearly presented. Typically the map function was not used to illustrate the importance of higher-order functions. Some candidates mentioned that such functions abstract away most of the control structures that are essential in imperative languages.
In answer to part b), candidates were expected to demonstrate functional programming specific knowledge, and to understand how environments are relevant to imperative programming languages. Candidates were asked to comment upon similarities, or otherwise, and should have discussed issues such as bindings, structures, signatures and functors. Typically candidates' knowledge of an environment was illustrated with the aid of an imperative language example. Whilst credit was given for this, discussions did not include any detailed reference to functional programming languages, and hence typically the issues such as bindings, structures, signatures and functors were not mentioned.
14% of candidates attempted this question, of whom 60% achieved a pass mark on it.

Question 4

a) What are the two major abstractions that characterise logic programming, and how are these compromised within the implementation of a practical logic programming language?

(12 marks)

b) Many languages have attempted to combine the advantages of logic programming with other programming paradigms such as object-oriented programming and functional programming. Present arguments for and against this integration.

(13 marks)

[Syllabus section 7e, Logic Programming]
In part a) of the question, candidates were expected to demonstrate knowledge of the basic concepts of pure logic programming. Based upon familiarity with a real logic programming language, candidates were expected to understand that 'extra-logical' features are required, which are not in keeping with a pure logical approach. The compromise discussion referring to logical or non-logical aspects, such as interface design or time dependency versus simplicity and consistency.
Candidates introduced the basic concepts of pure logic programming and the compromise with implementation issues such as specific search and order of computation rules. Typically candidates did not discuss other issues such as interface design, for example output statements, the implementation of numerical computation, and the use of the cut to improve efficiency, etc.

In answer to part b), candidates were expected to present 'for' arguments that might have included the very close relationship between the mathematics of functions (lambda calculus) and logic. The 'against' arguments were expected to be greater in number, and could have included differences in unification, use of inference engine for logic programming, etc.
38% of candidates attempted this question, 80% of whom achieved a pass mark on it.

Question 5

For a programming language to support concurrency, solutions to the problems of process synchronisation and communication are required. Elaborate on these problems, and describe the range of solutions that are available. In your answer, discuss the relative strengths and weaknesses of each solution.

(25 marks)

[Syllabus section 7f, Related Issues]
Candidates were expected to present a wide range of solutions including shared memory, semaphores, monitors, and synchronous message passing. To obtain high marks candidates were expected to demonstrate their understanding of the strengths and weaknesses inherent to these solutions. Additional credit was given for those candidates who were able to illustrate their answer with the aid of real concurrent language examples. Clarity and conciseness of exposition were also rewarded.
Whilst the candidates managed to provide a satisfactory overview of the problems referred to, in some cases the solutions they described were limited. However, a number of real concurrent problem examples were given, and a number of real concurrent language examples provided.
There were a number of excellent answers to this question and many answers provided suitably annotated examples and therefore attracted higher marks.
60% of candidates attempted this question, 96% of whom achieved a pass mark on it.