THE BCS PROFESSIONAL EXAMINATION
Professional Graduate Diploma

April 2001

EXAMINERS' REPORT

Programming Paradigms

This module enjoyed a 100% pass rate, although the number of candidates attempting was low. The highest mark was 79%. The general level of candidates' performance was therefore satisfactory.

QUESTION ONE

"When one first encounters a new programming language the question is usually: What can this language 'do'?"

(Ben-Ari 1996)

Choose two different programming language types (such as data-oriented, imperative, object-oriented) and:

a) Compare and contrast what each language type can do (15 marks)

b) Evaluate what type of applications they are most suitable for (10 marks)

When comparing the languages, the candidates often chose OO programming and something else. They tended to know the OO very well but were weak on the alternative. Often some implied the alternative was just the opposite of OO. Some didn't read the question and listed all three types of languages and not two. Some seemed to think that you couldn't use control structures such as loops, if/then/else etc within the definition.

Answers to section b) were weak. A typical answer was that OO for very hard applications and non-OO for simple things! Some candidates believed that concurrency was not achievable before the advent of OO!

Answer Pointers

a) The question is open to a variety of answers depending on which programming types have been chosen for the examples. The syllabus covers imperative languages (e.g. Fortran, COBOL, PL/1, Algol, C etc), data-oriented languages (APL, SETL) object-oriented languages (C++, Smalltalk, Java) and non-imperative languages (Lisp, ML, Prolog). A description of what the chosen languages were was expected. A good answer would have given concrete examples to show how they compare or contrast to demonstrate the candidates' practical understanding of the theoretical points discussed. For example, object-oriented (OO) languages support inheritance, encapsulation, object identity, which are not typically found in non-OO languages. Data-oriented languages such as APL and SETL have evolved from mathematical formulism, used to describe calculations.

b) A description of the characteristics of the chosen languages should have been given in part a). In this part candidates were expected to discuss what types of applications the language is most suitable for and a good answer would have related these characteristics to the application. For example, imperative languages are more general-purpose languages and could be used for a variety of applications. Whereas non-imperative languages include functional and logical languages and are targeted specifically at specialised programming applications. OO languages may suit a company who use OO methodologies for designing the system.

QUESTION TWO

The IT division of a large catalogue (mail order) company has traditionally used COBOL for developing and maintaining its Customer Order System. The system no longer fully meets the business needs of the company and a new system has been proposed. The Division is considering developing the application code in an object-oriented programming language.

With reference to an object-oriented programming language that you are familiar with, write a report that discusses the advantages and disadvantages of using such a language. Illustrate your answer with appropriate examples.
(25 marks)

Answer were adequate but some of them were shallow and candidates tended to cover only the advantages, failing to reflect on any disadvantages. Few examples were given.

Answer Pointers

Candidates were expected to consider the advantages and disadvantages of using an object-oriented (OO) language such as Smalltalk, C++ or Java. Their answer should have included a discussion of the concepts found in OO languages but not in a language such as COBOL. For example; inheritance, polymorphism, encapsulation, reuse, information hiding etc. They needed to reflect on what benefits or problems these concepts bring to the given environment. Examples from an OO programming language should have been included to illustrate the points made.

QUESTION THREE

a) Describe four important tools typically found in an Interactive Development Environment (IDE) (10 marks)

b) Discuss how these tools improve the productivity of programmers and the quality of the code they produce (15 marks)

For both parts, examples are expected of appropriate tools, for example, Microsoft Visual Studio, Borland C++, Borland Jbuilder etc. They need to identify what elements of the tool are useful to productivity and say why they think they are beneficial or not.

Answer Pointers

a) Candidates should have picked four tools available for developing a program typically found in an IDE and describe what they are. Candidates may discuss debuggers, testing tools, linkers, configuration tools, loaders, screen painters, code generators etc. Most candidates answered this section reasonable well but some lost marks because their description was sketchy and some because they classified debugging and testing as separate activities but then proceeded to describe them as if they were the same.

b) This part required the candidates to reflect on the real usefulness of these tools. Within their discussion they should indicate why they think the tools are important are not. For example, they could argue they improve the speed and performance of the programmer, help with debugging, testing, etc. Disadvantages could include that the programmer may fall into bad programming practices by relying on the tool to correct mistakes, rather than checking and testing it thoroughly themselves.

Most candidates were only able to think of benefits and advantages. The question of the quality of the code produced was not addressed.

QUESTION FOUR

a) Higher-order functions can be found within the implementation of applicative (functional) programming languages. Define the nature of higher-order functions and provide illustrative examples of their roles within a functional programming language with which you are familiar. (12 marks)

b) In addition to defining functions and evaluating expressions, a functional programming language can contain declarations. Discuss the nature of the declarations within a functional programming language and comment upon any similarities or differences that might exist between declarations in a functional and an imperative programming language. (13 marks)

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

Answer Pointers

a) In this part, candidates were expected to define the nature of higher-order functions, and to provide suitable illustrative examples of their role within a functional program. It was expected that the importance of such functions be discussed, in particular the map function, which should have highlighted a major difference between functional and imperative languages. Mention should have been made of higher-order functions and the map function and examples of its use. Also, that such functions abstract away most of the control structures that are essential in imperative languages.

b) In answer to this part, candidates were expected to demonstrate functional programming specific knowledge and to understand how environments are relevant to imperative programming languages also. Candidates were asked to comment upon similarities, or otherwise, and should have discussed issued such as bindings, structures, signatures and functors.

QUESTION FIVE

In order to support concurrency, a programming language needs mechanisms to handle process synchronisation and communication. Describe the problems to which this need gives rise and some of the solutions proposed, discussing their strengths and weaknesses. (25 marks)

Answer Pointers

Candidates were expected to present a wide range of solutions including shared memory and the mutual exclusion problem, 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 available to candidates able to illustrate their answer with the aid of real concurrent language examples. Whilst the candidates managed to provide a satisfactory overview of the problems referred to, their presented solutions were somewhat limited. All the candidates were weak when it came to demonstrating their understanding of the strengths and weaknesses inherent in their chosen solutions. Some real concurrent problem examples were given but little to no real concurrent language examples were provided.