THE BCS PROFESSIONAL EXAMINATION
Diploma

April 2000

EXAMINERS' REPORT

Software Development Environments

Question 1
Answer Pointers
This question examines section 1 of the syllabus, "The Software Development Lifecycle"

(a) Describe and distinguish between the requirements, specification and design stages of the software development lifecycle. For each stage, indicate a common error that could be made.

(8 Marks)

Requirements
- [Description] A list of features that should be present in the finished software.
- [Common Error] A missing requirement could lead to a system being built to specification but not what the user wanted.

Specification
- [Description] A list of (abstract) data structures & operations with implicit relationships between data & results.
- [Common Error] An inconsistent specification with a contradiction in it will lead to problems at a later stage.

Design
- [Description] A list of data structures and operations with explicit relationships between data & results (algorithms).
- [Common Error] Lack of attention to the detail of interface issues can lead to individual modules being designed correctly but used wrongly (e.g. called with transposed parameters).

(b) Where would you place the construction of a prototype in the software development lifecycle and how does it help with the early stages of software development?

(5 Marks)

An analyst creating requirements or specification documents may baffle a customer with notation in these documents; showing a prototype application is very direct but requires a huge amount of effort.

If there is a significant interface to the application then showing the customer the 'look and feel' of the interface (with no functionality behind it) can achieve much of the same communication with considerably less effort.

(c) ***Various diagramming notations can be used at the specification stage of the software development lifecycle. Describe one such notation and provide a simple example of its use.***

(7 Marks)

Description of a diagramming structure required
e.g. dataflow diagrams

with definitions of symbols for

- dataflow (directed)
- user interaction
- transformation process
- data source

together with a (simple) example.

(d) ***A specification can be said to be informal or formal. What is meant by a formal specification? Give an example of a situation where a formal specification would be used or preferred over a less formal specification.***

(5 Marks)

A formal specification is usually presented as a set of mathematically precise, hierarchical relations regarding acceptable data (pre-conditions) and results obtained from the data (post-conditions). If a pre-condition is satisfied by some data, then the program is required to produce corresponding output that satisfies a post-condition.

This kind of specification is necessary if a formal proof of correctness of the eventual program is required. This might occur in a safety-critical application.

Examiner's Guidance Notes

Several candidates read the actual phrase "requirements, specification & design" (with comma, 3 stages) as if it had been written "requirements specification & design" (no comma, 2 stages).

Candidates failed to describe the notation system itself, symbol by symbol.

Very few candidates seemed to know anything about formal specifications.

Question 2
Answer Pointers

This question examines section 2 of the syllabus, " The Programming Environment"

Apart from the facilities of the programming language itself, there are wider issues to be considered in choosing a high-level language for a project. These include the way the language is implemented (compiler/interpreter), the underlying operating system and the possibility of having to write in a platform-independent style.

(a) ***"A beginner writes a whole program in a single file and runs it using an interpreter. A professional constructs a set of source files and uses a compiler." Referring to this quotation, explain the difference in scale of the programs involved and the different demands made on the language implementation system used.***

(10 Marks)

Single file, interpreted:

- Simple
- Fast 'translation' time
- slow 'run' time
- suits learner

Project, compiled

- Sophisticated
- multiple source documents
- multiple object documents
- some from system libraries, some from programmer
- Slow 'translation' time
- fast run-time
- suits production program

(b) ***Under one operating system a program is supplied with data by dragging a data file onto the icon representing the program. In another the name of the program is typed followed by the name of the data file.***

Contrast the two approaches and describe scenarios where each is shown at its best.

(10 Marks)

WIMP style OS:
Dragging data onto program icon is the graphical, windows oriented style.
It is very easy to learn
and perform
and places no burden on the memory.

Command-line OS:
Typing a command and data is harder to learn,
harder to perform
and requires the user to remember the commands.

WIMP best for…
Drag & drop is well-suited to individual program executions because it is quickest.

Command-line best for…
For a repetitive task like "process all the data files in the directory that have filenames like x?y*", the command line structure using wildcard matching and/or script files is superior.

(c) What precautions should be taken when writing an application in a high-level language to ensure cross-platform compatibility?

(5 Marks)

- Use only the standard language (no extensions)
- Try not to rely on potentially machine specific details (e.g. word length, range of integers, etc)
- Isolate likely problem areas (e.g. graphical interface) to as small a section of the program as possible

Examiner's Guidance Notes

Statements like "a compiler is fast" or "a compiler is slow" were meaningless in the context of this question without being attributed to either compilation/translation time or run/execution time.

Candidates failed to pick up on the fact that this was an operating system question and spoke about the difference between older compiler environments and the newer visual environments.

Thinking about cross platform compatibility seemed to be outside the experience of many candidates.

Question 3
Answer Pointers
This question examines Section 5 of the syllabus, "Program Testing."

(a) Describe in detail the operation of TWO different tools that can be used to assist the testing process. Your answer should indicate the advantages and any disadvantages associated with each approach.

(13 Marks)

Any class of testing tool is acceptable, but typical examples (indicating the level of detail required) include:

Dynamic Analysers

- Provide information on how often each statement is executed
- Add statements to the code during compilation (or by using a pre-processor) to gather and collate the required information. (This is referred to as "Instrumentation")
- Results are then displayed at the end of the execution, typically numbering each statement and indicating how often it is executed.
- Typically, each loop, decision and block of sequential statements are instrumented.
- The two principal uses are to ensure each statement is executed at least once and to identify regions of the program to be considered for optimisation.

Disadvantages

- Rely on all of the source code being available - not always possible if pre-compiled libraries are used.
- Instrumentation code can affect timing, which may not be suitable for, for example, real-time systems.

Test Data Generators
- Automatically generate a large number of test inputs.
- Test inputs are generated using rules provided by the tester.
- Particularly useful when the performance of a system in a practical environment must be tested (e.g. a database management system).

Disadvantages
- The associated output for each test case needs to be created.

(b) Describe a method used to derive test cases. Illustrate how your selected technique would be applied to a routine that sorts integers held in a dynamic (linked) list.

(12 Marks)

Candidates would be expected to discuss a suitable technique, for example, equivalence partitioning:
- Attempts to minimise the number of tests conducted (reducing costs) while simultaneously minimising the likelihood of errors.
- Group possible inputs into classes that have some common property, e.g. invoke the same statements within the program.
- Test one input from each class and any special cases such as boundary values.
- Assume that if the test is successful for the selected input, the program will function for all inputs in the particular class.
- Does not "prove" a program correct, can only detect errors

Other suitable techniques are also acceptable.

Application to the sort routine:
Typical Groups
- Empty list
- Lists with a single item
- Lists with 2 items
- Lists with 3 or more items
- Lists containing positive and negative numbers

Examiner's Guidance Notes

Several candidates discussed testing "techniques" (such as black and white box testing) rather than "tools".
In general, the first part of part (b) was answered well. However, answers relating to the application of the selected technique to sorting the dynamic list tended to concentrate on the data held in the list rather than the list itself.

Question 4
Answer Pointers
This question examines section 3 of the syllabus "Program Design"

When describing a programming language it is convenient to distinguish between program structures (also called control structures) and data structures. Data structures can further be broken down into internal data structures and external data structures. The three parts of the question below explore program structures and data structures and some of the relationships between them. You should choose an appropriate language or languages with which you are familiar on which to base your answers.

(a) ***One kind of program structure is called a selection structure. For a two-way choice this usually involves the keyword "if". For a multi-way choice (more than 2) a separate structure can be used. Describe the syntax and operation of a specific multi-way selection structure.***

(8 Marks)

Would expect an answer based on e.g. Pascal **case** construct or C **switch**
Syntax (accept syntax diagrams, BNF or structured English).

<case_statement> ::= **case** <exp> **of** <case_arms> **end**
<exp> is any expression
<case_arms> is one or more occurrences of <case_arm>
<case_arm> ::= <const> : <statement>

Each <case_arm> gives a constant and a statement that is executed if the value of <exp> matches the constant.

There may be a catch-all <case_arm>, e.g. otherwise:<statement> to be used if none of the other <case_arm>s match.

(b) ***For any one programming language:***
(i) ***List the built-in data types;***
(ii) ***Provide a code fragment which shows the declaration and use of a programmer defined type.***

(10 Marks)

Name of language: e.g. Pascal
List of built-in types: integer, real, char, boolean
Anticipated constructed data types would be array or record

Array declaration:

```
var v:array[1..10]of char;
```

Array use:

```
for i:=1 to 10 do
   v[i]:='x';
```

OR

Record declaration:

```
var r:record f1:char; f2:real end;
```

Record use:

```
r.f1:='x';
r.f2:=2.0;
```

(c) The data structures external to a program are files of various kinds. Give advice to assist a programmer in choosing whether to use a 'text' file or a 'binary' file for a particular situation.

(7 Marks)

Text-file:

- Simple, built-in to core of language
- Easy to create test data (any text editor will do)
- Slow because of ASCII to binary conversions or vice versa
- Sequential access

Binary file:

- More sophisticated
- Normally needs a separately constructed program to produce test data for input - or to read/check a file of output
- Fast because little or no conversion takes place
- Usually offers direct, random access

Examiner's Guidance Notes:

The question clearly asks for the syntax and operation of the statement. Reduced marks were awarded to candidates who merely provided an example of the statement in use. Despite strenuous efforts on the part of the examiners in the wording of the question, some candidates still answered on the basis of an 'if' statement.

Some candidates did not name the chosen language. The declaration of variables of built-in types was offered when programmer-defined data types were required. A declaration of a variable was often given which was not followed by a use.

Candidates seemed to confuse the choice between text and binary output from a user program with the choice between interpreting and compiling programs.

Question 5

Answer Pointers

This question examines Section 4 of the syllabus, "Program Development Environments."

(a) Discuss FIVE typical compiler options that are commonly available when compiling a program. Indicate circumstances when each option would be used.

(8 Marks)

Examples of compiler options include:

- Select target platform (multi-platform installation)

- Read options from a file (Used when several options are to be selected)
- Optimise code for speed or size (Depending on implementation environment)
- Include/Exclude debugging code (Include debug when building for test purposes, exclude for shipping version)
- Define location of source directories/libraries and output directories (Useful when re-using library code)

Candidates are expected to indicate when and why each option may be used.

(b) Consider the following erroneous code fragment, which attempts to sum the numbers from 1 to 100 and forms part of a much larger program.

```
VAR
        Total : INTEGER
        Sum : INTEGER

Total = 9
FOR Sum = 1 TO 100 DO
        Total = Total + Sum
END
Print Total
```

Assuming the complete program contains no other references to the variable "Total", and that the code correctly compiles and links, state the logical error which is the cause of the erroneous output. Explain how a debugging tool could be used to assist a programmer in finding the error.

(7 Marks)

Error: The incorrect initialisation of the variable "Total".

Use of a debugger:
- Erroneous output concerns variable "Total", so place a "break point" just before the variable is used.
- Run the program to the break point and add a "watch" to observe the value of total.
- Stepping through the code the programmer will observe that the variable "Total" does not contain the correct initial value.

(c) Discuss FOUR features typically found in a language specific editor.

(10 Marks)

Typical features that should be discussed include:
- Colour coding of keywords, identifiers etc
- Automated code layout
- Ability to indent selected portions of text
- Automatic line numbering
- Ability to provide multiple or split windows
- Ability to fold (hide) portions of the program
- Syntax directed editing, where for example, the system automatically completes statements by adding "END's" where appropriate etc.

Examiner's Guidance Notes:

Some candidates explained the compilation process rather than directly answering the question. However, those that did address the question answered well.
Most candidates answered this particular part (b) of the question well, although some did indicate that the source of the error was syntactical (e.g. a missing "BEGIN.

Question 6
Answer Pointers
This question examines Section 6 of the syllabus, "Quality Assurance and Documentation."

(a) A newly established software development company has two people in its development team and concentrates on the production of a single, evolving product. The two developers work largely in isolation on different sections of the software, writing in their own individual programming styles. When a new feature or change to the existing program is required, one of the programmers updates the code and simply copies the modified files to the other developer's computer so that both programmers always have the complete and latest version. When orders are placed, the current versions of the files are copied directly from one of the developer's computers. The only documentation is the code itself.

Discuss the main problems associated with this situation.

(12 Marks)

Problems to be discussed should include:

Lack of quality control mechanisms
There is no defined quality procedures, such as code reviews or independent testing. Thus it is likely that the code will contain errors.

Lack of overall software structure
The product evolves as the developers get new ideas. There is no overall architecture for the software into which new functional units can be placed. Ultimately this will lead to a product that is difficult to maintain and update. A lack of structure is also likely to hinder code re-use.

Lack of documentation
There is no documentation to indicate the structure of the software or how it operates, implying that maintenance and future enhancements will be more difficult. (For example dependencies between functional units are not recorded.)

Lack of standardisation
Different coding styles, for example, indentation, variable names, commenting styles etc would further hinder maintenance and future changes.

Lack of version control
Each customer can potentially have a different version of the software, making product support and upgrades difficult.

Reliance on individual members of staff
One person has complete responsibility for each functional unit of the software, including its design, implementation and testing. Coupled with the lack of documentation, problems will arise if a developer leaves the company.

(b) The company is initially successful and hence employs a further two developers to enable additional features to be incorporated into the software. However, as the product increases in complexity and the size of the development team grows, it becomes apparent that a quality control strategy needs to be put in place. Assume you have been employed as an external consultant.

Write a report outlining a suitable strategy. Your report should highlight the benefits of implementing your proposal.

(13 Marks)

The report should discuss the following:

- Produce Company coding style, including commenting, use of variable names and indentation.
- Document the existing software (e.g. define what each functional unit does, its inputs and outputs, any relationships with other parts of the software, for example, updating global variables, the original author etc). An overall system architecture should also be identified as part of this process.
- Define how new features and changes to existing code are to be documented. (e.g. For new features the system architecture should be updated and the information identified above should be recorded. For changes, the original documentation should be augmented with details of the change, why it was changed, the name of the person who made the change and the date the change was made.)
- Introduce some form of code review to ensure standards are adhered to. The review should also examine any associated documentation.
- Introduce independent testing where someone tests the code other than the developer, e.g. an end user. Documentation should be maintained to record errors discovered by the tester and to demonstrate that the developer has fixed them.
- Introduce a suitable version control strategy. This should include a strategy to ensure that two (or more) developers cannot make simultaneous updates to the same functional units and that all developers access the most recent versions of each file. This could be achieved by using, for example, a product such as Visual SourceSafe.

Benefits of implementing this strategy:

- Company coding style can be used when inducting new staff.
- Documentation of code structure and operation reduces dependence on individual staff members.
- Documentation facilitates maintenance and future enhancements.
- System architecture may identify areas of potential re-use, reducing future development times.
- Code reviews and the use of independent testing is likely to reduce the number of errors in the shipped product, increasing its reliability.

Examiner's Guidance Notes:

This question was well answered in general, although several candidates provided limited answers.

Candidates tended to state what should be done without any implementation details. For example, several candidates mentioned the need for a version control strategy but did not indicate how it might be implemented. Also, many candidates did not clearly indicate the benefits to be gained from their proposed strategy.