THE BCS PROFESSIONAL EXAMINATION
Diploma
April 2003

EXAMINERS' REPORT

Software Development Environments

General

- It is necessary for the examiner to be able to read your answer. Try to write legibly. In many cases writing less, but more clearly, would gain more marks.

- Candidates should answer the question posed. There was a strong tendency to simply identify key words in the questions and then regurgitate large quantities of textbook material on these topics rather than applying this knowledge to the question asked. Much if what was written by many candidates was simply irrelevant to the question, even if accurate.

QUESTION ONE

This question examines section 1 of the syllabus, "The Software Development Lifecycle".

A requirements analysis document contains the following, "After throwing five dice, decide whether all of them show a different number".

By using this requirements statement, demonstrate your understanding of the principle constructive steps of the traditional lifecycle process by providing

(i) a specification, (7 marks)
(ii) a design and (8 marks)
(iii) coding. (10 marks)

You may use whatever notations and languages you feel appropriate.

Answer Pointers

The question is designed to see if the candidate can write actual examples of specification, design and code for a specific example and make them look different.

(a) specification - example
if the five dice values are d1,d2,d3,d4,d5 then the specification of exactly four numbers are the same could be expressed by gathering the five numbers into a set and then checking whether the size of the set is 5

$$|\{d1,d2,d3,d4,d5\}|=5$$

(b) design - example
- store counts of numbers in array with subscripts 1 to 6,
- initialise elements to zero,
- record occurrences of each number in array as they are read.
- At end go through array, increasing variable size by one for every non-zero entry.

- Look to see if size is 5

(c) code - example

```
for(i=1;i<=6;i++) /*initialize a 6 element array*/
   count[i]=0;
for(d=0;d<5;d++){ /*read in 5 dice values, saving information in array*/
   scanf("%d",&v);
   count[v]++; /*e.g. if dice value is 4 then count[4] is increased by 1*/
}
size=0;
for(i=1;i<=6;i++)  /*go through array counting non-zero entries*/
   if(count[i]>0)
      size++;
if(size==5)  /*the size of the 'set' (see specification)*/
   printf("all different");
else
   printf("not all different ");
```

Mark Breakdown

(a) 7 marks – for specification, something that is a statement of the problem with no hint of data structures to be used or algorithmic detail

(b) 8 marks – for design, something that gives some idea of variables, data structures, together with an overview of the algorithm

(c) 10 marks – for final code, in some language, capable of reading five numbers and printing different messages depending on whether the numbers are all different

This question was misunderstood by many students. The question provides a scenario (a requirements statement) concerning dice and invites the student to write an actual specification, design and corresponding code.

Many answers just contained general descriptions of three stages of the software development life cycle without ever mentioning the word dice.

QUESTION TWO

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

(a) Many text editors have global search and replace facilities. Describe a programming situation where this facility might be used to good effect.
(4 marks)

(b) Instead of exact matches, many systems allow pattern matching. In this case the search is based on a pattern (or regular expression) using special characters e.g. '.', '?', '*', '+', '[ ]', '\'. For a system that you are familiar with, list the special characters and explain the effect of these special characters in patterns. **(12 marks)**

(c) Use your special characters to make patterns (regular expressions) that match
(i) any integer (any sequence of decimal digits)
(ii) any variable name (any letter, optionally followed by letters and digits)
(iii) any jpeg file name inside double quotes, e.g. "image.jpg"
(9 marks)

Answer Pointers

This question is based on regular expressions which are found in editors, programming languages, command languages and operating systems.

(a) global search & replace can be used to systematically change an identifier (of a variable, procedure, ...) if, for example, a naming convention has inadvertently been overlooked

(b) meaning of special characters in patterns

.	matches any character
x?	matches x optionally
x*	matches any number of x's including none
x+	matches any number of x's (at least 1)
[x-z]	matches any character between x and z inclusive
\x	turns off the special effect of x

(c) patterns to match…

any integer [0-9]+

the pattern consists of
- any digit from 0 to 9,
- occuring at least once

any variable name [a-zA-Z][a-zA-Z0-9]*

the pattern consists of
- any letter, lower or upper case
- followed by any alphanumeric character
  - occurring zero or more times

any jpeg filename in quotes ".+\.jpg"

the pattern consists of
- a double quote character
- any character
  - occurring at least once
- followed by an actual full-stop
- followed by the letters j, p, g
- followed by a double quote character

Mark Breakdown

(a) 4 marks for a reasonable example
(b) 2 marks for each explanation of a pattern character, total 12
(c) 3 marks for each required pattern, total 9

Some students misunderstood the question and wrote answers describing where ".", "*", were used in general programming languages (decimal point in numbers, multiplication symbol). Those students who understood that the question was about regular expression notation obtained good marks.

QUESTION THREE

This question examines section 3 of the syllabus “Program Design”

(a) Construct a design which is structured using a combination of modularisation, sequencing, selection and iteration for the program which is described below. The design can be in any notation, diagrammatic or text, but your answer must be clearly labelled to show the type of structures used in each part.

(18 marks)

The program: “Read in 10 positive numbers, noting the maximum value, and then go through all the numbers and for each one print ‘+’ if it is greater than half the maximum and ‘-‘ otherwise.”

(b) Describe the syntax used to provide modularisation, sequence, selection and iteration in a programming language of your choice. [If your chosen language has more than one syntax for a particular control mechanism then you only need to describe one].

(7 marks)

Answer Pointers

The question tests whether the candidate can demonstrate the use of structuring by modularisation, sequencing, selection and iteration in a simple example.

(a) Expect diagrammatic answers (flowcharts?)

Module 'HalfMax' broken down into
  sequence of submodule 'Read10SetMax'
  followed by submodule 'MoreThanHalf'

'Read10SetMax' **module** broken down into
  sequence of set max to 0
  followed by iteration
    iteration over i from 1 to 10
      sequence read next number into n
      followed by set a[i] to n
      followed by selection
        selection if n>max, set max to n

'MoreThanHalf' **module** broken down into
  iteration over i from 1 to 10
    selection if a[i] > max/2, print '+',
          otherwise print '-'

(b) syntax for design structures, in C
modularisation via functions
declaration:
        fn_type fun_ident (formal_parameters){ body }
call:
        fun_ident (actual_parameters)

sequencing - no special token - just juxtaposition

selection: **if**(exp)stmt or **if**(exp)stmt **else** stmt or **switch**(exp){...}

iteration: **while**(exp)stmt or **do** stmt **while**(exp) or **for**(...){...}

Mark Breakdown

(a) For a design that breaks program down into 2 (or more) modules to be run in sequence – 6 marks.
For the design of the sequencing, selection, repetition of each of two modules - 6 marks each.
Total 18

(b) Two marks for the syntax of modularization, selection, sequencing structures, one mark for sequencing, total 7 marks

In part (a) the design of a program to read in 10 numbers and print out the messages was reasonably done. A few students tried to achieve everything inside one loop when clearly two loops, one after the other are required.

Most marks were lost by not "clearly labeling the structures used", i.e. drawing attention to where modularization, sequencing, selection and repetition were used.

For part (b) examples of the structures did not gain as many marks as syntax, e.g.

```
if ( <expression> ) <statement>
```

is worth more than

```
if (x>1) s++;
```

QUESTION FOUR

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

(a) Describe the facilities available within a program development environment with which you are familiar that would facilitate the implementation of a large system by a team of programmers.

(5 Marks)

(b) Discuss FIVE features often found in program development environments that assist individual programmers to develop code.

(10 Marks)

(c) Errors are often evident when modules written by individual programmers are brought together to form the complete system. Indicate how the use of a debugger can assist in the process of identifying the location of these errors.

(10 Marks)

Answer Pointers

(a) Code can be split across files (different modules), each of which can be individually edited and then compiled to validate syntax. All modules that form a complete system can be linked together under a "Project". When the project is "built" only those modules that have changed since the last compile are re-compiled, then the linker forms the final system.

Tools, for example, to assist version control, locking of modules being worked on by other developers and central data dictionaries are often provided.

Some candidates discussed environmental issues, such as adequate air conditioning, each developer having their own computer, or other external factors including briefing meetings and good teamwork skills. However, these are not provided by program development environment and hence could not be considered as valid answers to the question.

(b) Examples include (but are not restricted to):

Auto completion of code
For example, when a programmer types the first few characters of a keyword, the environment completes the rest

Automatic indentation
Body of functions, loops etc indented according to predefined rules.

Colour Coding
Highlights keywords, comments etc and helps identify typographical errors such as spelling or missing closing comment token.

Syntax Directed Editing
Automatic insertion of closing tokens when opening token entered (e.g. BEGIN/END combinations).

Context Sensitive Language help
Provides help on the keyword/function under the cursor, e.g. indicating the nature of parameters required for a function.

In general this part of the question was well answered.

(c) Candidates were expected to discuss the use of he following features.

Break points
User defined locations at which the debugger will pause execution of the code until further instructions are given.

Watches/Inspectors
Display the contents of a variable or a more complex expression when the program is paused at a break point.

Changing variable contents
Allows the content of a variable to be modified during the execution of the code.

Stepping into/through code
Allows the code to be executed statement by statement. Statements that represent calls to other functions and/or procedures can either be considered as a single statement (step through) or as a group of separate statements that get executed individually (step into).

Call stack
Presents information as to which functions/procedures have invoked other procedures.

Again, in general, this part of the question was well answered.

Mark Breakdown

(a) 1 mark for each point made, up to a maximum of 5 marks.
(b) 1 mark for each feature identified, 1 for associated description up to a maximum of 10 marks.
(c) 1 mark for each feature identified plus 1 for its description, maximum 10 marks.

QUESTION FIVE

This question examines Section 5 of the syllabus, "Program Testing".

(a) Compare and contrast TWO structured testing methods with which you are familiar, highlighting the advantages and disadvantages of each.

(12 Marks)

(b) Using a method with which you are familiar, list a suitable set of test cases for the pseudo-code below. Candidates should also clearly explain how the test cases were derived.

```
Read values of x, y, a and b
if (x>y) then
  if (a<b) then
    c=0
  else
    c=99
else
  if (x==y) then
    c=50
  else
    c=25
Print c
```

(8 Marks)

(c) Identify the advantages and disadvantages of a testing tool with which you are familiar. Provide an example of where it would be appropriate to use the tool.

(5 Marks)

Answer Pointers

(a) Candidates should discuss, for example, top-down versus bottom-up testing or black versus white box testing. The following sample solution is indicative of the detailed required.

Black Box

Testing is based on the requirements specification and without reference to the actual code. Test cases are selected to ensure that all of the requirements are achieved in that particular inputs generate the required output or anticipated response.

Advantages

- Testing can be conducted by the end user
- Tester and programmer are independent
- Will help to expose any ambiguities or inconsistencies in the specifications
- Test cases can be developed as soon as the specification is complete

Disadvantages

- Cannot ensure that all sections of the code have been tested.
- Certain types of error may not be detected, e.g. two errors that combine to cancel each other out.
- Cannot be directed toward specific segments of code which may be very complex (and therefore more error prone)

White Box

Testing is based on the actual code. Test cases are selected to ensure each path (i.e. each branch of a conditional statement) is executed at least once.

Advantages

- Each individual line of code will be tested at least once.
- Beneficial "side effects" often arise, e.g. code optimisation
- Will detect a wide range of errors
- Can be focussed on specific areas of the code, e.g. areas containing complex logic and thus more likely to contain errors

Disadvantages

- Testing does not ensure program satisfies the requirements (It will miss requirements missed in the code)
- Test cases must be developed by a skilled employee with knowledge of computer programming
- Test plans cannot be developed until the code is complete

In general this part of the question was well answered.

(b) Any technique is acceptable, solution below uses equivalence partitioning:

Step 1 - Divide all possible inputs into classes that produce similar actions within the program. Classes for given code:

(i) Enter values of x > y and a < b
(ii) Enter values of x > y and a >= b
(iii) Enter values of x = y, any value for a and b
(iv) Enter values of x < y, any value for a and b

Step 2 – Identify any additional boundary cases

(v) Enter values of x > y and a = b

Step 3 - Select 1 test from each class
Possible values:

	x	y	a	b
(i)	4	3	2	5
(ii)	4	3	5	2
(iii)	5	5	1	1
(iv)	3	5	3	2
(v)	4	3	1	1

Most candidates were able to identify the majority of the test cases required, but few "clearly explain[ed] how the test cases were derived".

(c) Any testing tool is permitted; solution below is for a simulator:

Typical Use

- Safety critical software or where damage to associated hardware could be very expensive.

Advantages

- Can be more cost effective if interfacing hardware is expensive
- Errors in the software do not invoke catastrophic events

Disadvantages

- Could be errors in the simulator
- Difficult to test functions that are dependant on timing

Many candidates failed to "Provide an example of where it would be appropriate to use the tool", but otherwise the question was, in general, well answered.

Mark Breakdown

(a) 2 marks for a description of **each** technique and 4 for its respective advantages and disadvantages (1 mark for each advantage/disadvantage identified up to a maximum of 4).

(b) 5 marks for the actual test cases, 3 for the description of their derivation.

(c) 1 mark for a typical use of the tool, 2 marks for advantages and 2 for disadvantages.

QUESTION SIX

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

(a) Identify FIVE factors used to indicate the quality of software. For each factor, indicate what would be considered "high quality".

(10 Marks)

(b) Standards can be used to ensure consistency of the code produced by different programmers. Describe how a coding standard with which you are familiar addresses the following aspects:

(i) Naming of identifiers
(ii) The use of comments
(iii) Indentation

(6 Marks)

(c) Most commercial software gets modified frequently after the initial development phase. Describe how the process of deciding which requests for changes should be implemented could be managed within an organisation that receives many requests for such updates.

(9 Marks)

Answer Pointers

(a) Any five quality measures are acceptable, for example:

Usability
Good "usability" implies software that conforms to standard HCI guidelines.

Maintainability
Software is maintainable if it is well documented (both in the code through comments and through supporting external documentation) and, where different alternative algorithms were available, the simplest chosen.

Reliability
Software is reliable if it always produces the correct results.

Robustness
Software is robust if it handles error situations in a controlled manner (e.g. invalid inputs)

Efficiency
Software is efficient if it uses as little memory as possible and secondly if it uses the smallest number of processor cycles possible.

In general, this part of the question was well answered.

(b) The answers below are indicative and not the only possible solution.

Naming of identifiers

- The initial characters of the identifier should indicate its type (e.g. “int” for integer, “str” for string etc.)
- Each “word” within a name should commence with a capital letter, all other letters should be in lower case, e.g.intCountOfApples.

The use of comments

- Each module should have a comment block at the start indicating the name of the module, the original author, a description of what the module does and a list of changes to track different versions. Changes should be tagged with the date and author making the change.
- All variable names should have a comment indicating their use.

Indentation

- Position of tab stops should be every 3 spaces
- Specific rules should be defined for each syntactic structure

Many candidates struggled with this question with many concentrating on the benefits of having naming conventions, good commenting standards and indentation rather than “describing” a coding standard for each factor.

(c) The solution below is indicative of the sort of answer required and is not the only possible one.

Requests for changes should be submitted on a standard form to include the following information:

- Name and contact details of person making the request
- Date of request submission
- A checkbox indicating whether the request relates to an error in the existing software or a request for additional functionality
- A description of the requested change

Bug fixes go to the support/maintenance team

- A check is made to see if the bug has previously been reported. If so, the additional report form is filed with the others already received.

- New bugs are allocated a priority level (say 1-5) depending on (for example) the severity of the problem, the number of customers impacted, the time required to fix the fault etc. If additional forms are received reporting the same bug, the priority may be changed.

Requests for new functionality go to the development team

- Again a check is made to see if the same or similar request has been made previously and if so, the form filed with the previous ones.
- Again each form is allocated a priority level depending on (for example) the number of customers likely to benefit from the new functionality, the amount of time required to implement the new feature. Additional forms requesting the same change may alter the priority assigned.

Forms are filed based on the module or area of the program's functionality that is impacted.

Issues with the highest priority are implemented first.

However, need to ensure that request with low priority are not permanently ignored (particularly bug fixes) and hence the length of time in the "queue" may increase the priority of a request.

This part of the question was poorly answered, with many candidates not even attempting an answer. Candidates should expect questions that require them to apply their knowledge to practical applications of software engineering and not just rote learn material.

Mark Breakdown

(a) 1 mark for each measure, 1 for description of high quality, maximum 10 marks.
(b) 2 marks each for identifier naming conventions, commenting style and indentation.
(c) 1 mark for each point identified.