THE BCS PROFESSIONAL EXAMINATION
Certificate

April 2001

EXAMINER'S REPORT

Software Development

QUESTION ONE

1. Code cracking- simple codes, based on one arbitrary symbol (character) for each letter of the alphabet, could be broken easily by means of frequency tables. Thus in the English language 'e' and 't' greatly predominate in any text, followed by 'n','r','o', ect. Thus a count of the different symbols in a simple coded message can be made and the one occurring most frequently can be assigned as 'e', the next 't' and so on. Other letter frequencies in sufficiently long messages are also close to the values given in the frequency table (see Table 1.1) at the bottom of the page.

a) Write an algorithm in pseudocode, for a well-structured program, where an input text file 'text' containing a simple coded message is to be read. Counts are to be made for all the symbols read, ignoring all spaces and punctuation (if present). A sorted list of the frequency of each symbol is then taken to be printed. The algorithm should then use the frequency table (Table 1.1) to assign letters to the symbols in descending order of occurrence. The original coded message is then to be out and beneath it are to be printed the corresponding assigned letters and an '*' for the rest.

(24 marks)

b) Discuss the data structures which would be needed before the program could be developed further.

(6 marks)

You may assume that the procedure SORT() with appropriate parameter is available for your use. Code for a sorting technique is not required.

Example with three assigned letters (e,t,n) where the decoded message is
The BCS examination is dreadfully difficult this year :-

…………………….coded message……………………

```
t*h *** e****n*t*n ** **e******* ********t t*** *e**
```

E	13.0		…	…..		U	2.6
T	9.2		S	6.1		M	2.5
…	…..		D	4.2		…	…..
N	7.9		L	3.6		X	0.5
R	7.6		H	3.4		Q	0.3
O	7.5		C	3.1		K	0.3
A	7.4		F	2.8		J	0.2
I	7.4		P	2.7		Z	0.1

General Comments

There were very few answers to this question and most answers were of a poor quality. The problem is relatively straight forward but candidates did not appear to have the ability to extract the key stages and then break each stage down into a little more detail.

Guidelines to Answer

The problem is basically

```
Initialise data structure for counts of LETTERS
Until end of text file do
    Read each character in turn
    Write character
    If a LETTER then
        Update count for character read
    Else
        Skip
    End if
End do
Write the end of line
Sort character counts into descending order
Assign top character as ‘e’, next as ‘t’, etc
Reset input text file
Until end of text file do
    Read each character in turn
    If a LETTER then
        Look up assigned letter
        Write letter
    Else
        Write ‘*’
    End if
End do
Write the end of line
```

One or two refinements of the algorithm giving a little more detail in each refinement was sufficient.

A data structure capable of holding each letter, with its count and its equivalent decoded letter was necessary. A set, or equivalent, for the acceptable letters would be helpful.

QUESTION 2

a) **Dry run the algorithm given below (Algorithm 2.1) with suitably chosen test data values. Only one dry run in expected. ‘max’ should restirct to 3 or 4.**

(16 marks)

b) **Explain the purpose of the algorithm. Highlight any problems you believe the algorithm contains.**

(6 marks)

c) **Translate the algorithm into a suitable procedural language. Add appropriate comments and input prompts and more meaningful output statements. State the language used.**

(8 marks)

```
Line
No      Algorithm

0.      DECLARE data as ARRAY [1..max]
1.      Total <- - 0
2.      INPUT limit
3.      FOR ct <- - 1 TO limit DO
4.      BEGIN
5.              INPUT data[ct]
6.              total <- - total + data[ct]
7.      END
8.      Average <- -total/limit
9.      PRINT average
10.     Above <- - 0
11.     FOR ct <- - 1 TO limit DO
12.     BEGIN
13.             IF data[ct] GRATER THAN average THEN
14.             BEGIN
15.                     above <- - above +1
16.                     PRINT above, data[ct]
17.             END
18.     END
19.     PRINT ABOVE
```

Algorithm 2.1

A variety of notations for the dry run were used and accepted. Most candidates were able to determine that the algorithm was intended to find the average of the items input and then write out those above the average

Guidelines to Answer

If a column method was adopted for the answer, e.g.

Line number	total	ct	data[ct]	average	above	Comment
1	0	?	?	?	?	
2						4 -> limit
3,4		1				
Etc						

then relevant column headings were looked for.

A potential problem with the algorithm was an inconsistency between ‘max’ and ‘limit’.

A range of languages was used. Appropriate comments along with meaningful input prompts and output statements were looked for.

QUESTION 3

a) **What features should be incorporated into a good CASE tool and why?**

(11 marks)

b) **Critically comment on ONE CASE tool with which you are familiar.**

(11 marks)

c) What impact do you believe the continuing development of the World Wide Web will you have on CASE tools and why?

(8 marks)

General Comments

This question was not well answered. Answers to part (a) rarely gave four features and four rationales. Features were often given without rationale and features not clearly linked to rationale.

Many candidates offered techniques and diagramming conventions (including flow charts) as examples of CASE tools. Many gave good examples but failed to adopt a suitable critical stance and simply listed the features.

Answers to part (c) likewise often omitted rationale, or struggled to get more than one impact and explanation.

Guidelines to Answer

Features that candidates could have mentioned include

- WYSIWYG - screens and reporting for verifiable product
- enforce standards - for maintainability and change management
- easy code generation - for productivity, complexity management, design transparency, requirements tracing, architecture flexibility
- triggers - for customisation
- interoperability - for deployment on multiple platforms
- GUI - robust interface (for users)
- version control - for management, complexity management
- repository - holding project's essential materials
- reverse engineering - recovering design documentation from code

Candidates were expected to critique a CASE tool using the framework of features introduced by the candidate in part (a).

Possible impacts include matters such as market growth leading to cheaper, downloadable tools, new tools to address needs of web such as security, media content, interaction, and new processes such as distributed development teams (internationally, work-at-home, etc.) or novel web business ideas.

QUESTION 4

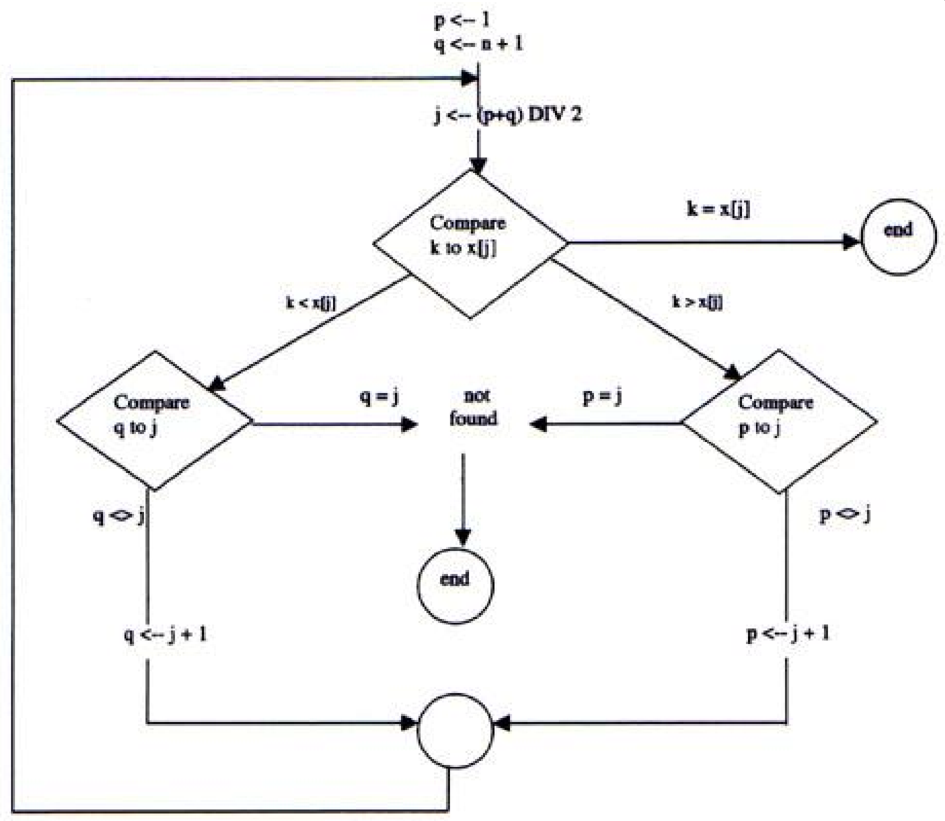
The binary search technique is important in locating a single record from a table held in an array (x[j] for j = 1 to n) using a single key (k). A flowchart, taken from an old computing book of methods, is given below:

a) Translate this flowchart into pseudocode. Meaningful name for the items coded should be used.

(18 marks)

b) **Convert this pseudocode into a structured form which does not use 'GOTO' statements and which would be suitable for straightforward translation into a procedural language. Include comments and prompts for the input of data as required.**

(12 marks)



'p is the lower bound of the search	'J' is the current position being tested
'q' is the upper bound of the search	'k' is the required key value
'n' is the size of array 'x' ot its upper bound	'x[j]' is the data item currently being compared

'DIV' means inter division: e.g. 7 DIV 3returns 2

General Comments

This was a practical type of question, as opposed to theoretical, and had a disappointingly low average. In general the practical answers in this paper were weaker than the book work/recall type of question.

Guidelines to Answer

The pseudo code was a simple loop with conditional statements. GOTOs were acceptable in part a). The quality of the candidates' pseudo code affected how easily the candidates were able to convert their pseudo code into a structured form not requiring GOTOs.

SECTION B

QUESTION 5

Selection, sequence and iteration are comon constructs in programming languages. Show, via examples, how these constructs are incorporated into a design method of your choice. (12 marks)

General Comments

This was a standard book question with the slight addition of asking the candidate how the constructs fitted into a design method.

Guidelines to Answer

A simple example of each construct showing its meaning was required. The next stage was to show how these constructs fitted into a design method. The majority of candidates found the simplest way of achieving this was via a top down diagramming method.

QUESTION 6

The constructs 'function' and 'procedure' are used extensively in programming languages.

a) Describe these construct using examples. (6 marks)

b) Describe a context where either a function or a procedure could be used. (3 marks)

c) Give an example where it would be more appropriate to use a function. (3 marks)

General Comments

A number of candidates had difficulty in explaining the difference between a function and a procedure. A common mistake was not realising that a function returned a value and could be used in a statement such as

Variable := func (…).

Guidelines to Answer

The first part required just a brief explanation of each term.

A situation where just one value was required to be returned was acceptable for part b).

A simple example such as evaluating a factorial (or the example in question 7) with a brief explanation of why a function was appropriate was all that was necessary for this part of the question.

QUESTION 7

If A is an approximation to the cube root of a real number N, then A + C is a better approximation where C is given by the formula

$$C = \frac{1}{3}\left[\frac{N}{A^2} - A\right]$$

a) Write a function which takes two parameters N and E (where E is a given error value) and which repeatedly evaluates the cube root of N until subsequent values differ by ledd than the chosen value, E. the first approximation of the cube root (A) should be the square root of N.

(10 marks)

b) Why might this function never terminate?

(2 marks)

General Comments

Most candidates were able to code up the function. An element not always recognised was that the result might oscillate so it was preferable to take the absolute value of the difference.

Guidelines to Answer

It was important for candidates to remember to use real (as opposed to integer) values for the evaluation.

The most likely reason for the function not to terminate is if the error value selected is too small and the difference never satisfies the end condition.

Question 8

Compare and contrast, with examples, the following pairs of terms:

a) compilers and interpreters (4 marks)
b) stacks and queues (4 marks)
c) white box and black box testing (4 marks)

General Comments

For this question candidates had to recall, as opposed to apply, information and as a consequence was answered reasonably well.

Guidelines to Answer

a Compilers - translate prior to execution; can have compiler error propagation
Interpreters - translate by parts and execute as translation occurs; detailed error conditions possible

b stack – Last In First Out; application in parsing and command-line interpreting
queue – First In First Out; application is OS device management, buffers, etc

c white box - test of paths by inspection of code
black box - functional testing from requirements spec, using boundary values, stress testing, and other forms of requirements analysis to generate test data.

QUESTION 9

Define, with reasons, a set of criteria suitable for assessing user interface design. Include in your answer the type of user you are considering.

(12 marks)

General Comments

Many candidates gave imprecise, discursive answers of untestable and unmeasurable 'criteria'. Many others gave lists of criteria out of context and unsupported by rationale. Some identified a user context then proceeded to ignore it.

Guidelines to Answer

The candidate was expected to clearly identify the user under consideration. Three criteria (defined as testable and measurable) for identification were required and the rationale for these criteria in the given user's context was also required.

Criteria that were expected included

- pull-down menus, scrolling text, mnemonic keys - - criteria for giving user control
- push buttons, scroll bars - criteria for using real-world metaphors
- menu navigation, scrollbar navigation, dialogue boxes, - criteria for 'natural' interfaces
- popup menus, option menus, selection check boxes - criteria for consistency of design
- changing cursor to a pointer or hour glass and changing colours as signals for communication of events and processes to the user
- avoidance of confusion from overlapping boxes/buttons

QUESTION 10

Write brief notes on each of the following:

a) library routines (4 marks)
b) independent compilation (4 marks)
c) parallel processing (4 marks)

General Comments

This question was not well answered. In particular, very few candidates gave convincing descriptions of independent compilation.

Guidelines to Answer

a library routines should include pre-written, pre-compiled, defined interface, defined calling convention, defined error codes for incomplete or otherwise failed action, and assistance to productivity and testing.

b independent compilation should contain compilation by parts, pieces later to be linked together, productivity from multiple module production, and sequence dependencies of compilation for interface checking

c parallel processing should include multiple processors, special compiling for splitting programs, description of types of parallel architectures and specific types of application (e.g. matrix calculations for weather prediction)

QUESTION 11

Identify the potential types of users of software documentation and the reasons why they need the documentation. By what methods might the users gain access to the documentation. (12 marks)

General Comments

Many answers gave descriptions of documentation content without rationale of need. Many discussions of access were extremely rudimentary and unimaginative.

Guidelines to Answer

There are a number of possible users of documentation. Candidates who gave full answers for three users gained full marks.

type of user	**description of documentation**	**access issues**
end-user	HCI controls, functions and features	on-screen, manuals
maintainer	code, test data and results, architecture design	microfiche, CD, manuals, disks
developer	requirements, architecture, detail design, code	microfiche, CD, manuals, disks
co-developer	platform, interoperability	microfiche, CD, manuals, disks, Internet
QA	loop-closing quality records at all lifecycle stages	microfiche, CD, manuals, disks

QUESTION 12

What are the main functions of the system software on a general purpose computer? State, the reasons, whether these functions differ significantly between a Personal Computer (PC) or a Linux/Unix type environment. (12 marks)

General Comments

The answers to this question were mostly unfocussed, taking a discursive approach rather than an analytical approach (as in guideline below). As a consequence, they were very narrow and found it difficult to reach higher marks for what was essentially a 'count them up' answer. Best answers gave a description of function and analysis of Unix boxes.

Guidelines to Answer

Main functions include CPU management, memory management, disk file management and I/O or network management.

Two broad approaches were acceptable. One approach was to note that Linux/Unix systems were not significantly different to PC systems because the machines under consideration were all Von Neumann architectures and therefore possessed the same basic needs. Alternatively, an acceptable answer was a convincing case that Linux/Unix machines were often employed in demanding, network-support roles that required unusual elements of system software support such as remote diagnosis, mirror disks and/or hardware redundancy/majority voting on breakdown. A major feature looked for was that the candidate could justify their answer.