BCS PROFESSIONAL EXAMINATIONS
BCS Level 4 Certificate in IT

April 2008

EXAMINERS' REPORT

Software Development

General Comments

There continues to be approximately 10% of very weak candidates who get 10% or less. Many scripts were suggestive that they have not understood any of their course or they did not attend one.

Algorithmic development continues to be weak for most candidates

Candidates should also learn to answer questions without excessive rough work; this takes up valuable time.

Declarations made in answer to one part of the question (e.g. a record) need not be copied out again if the declaration is subsequently used in another part; no extra marks are gained.
System directives like [uses crt, clrscr, #include <iostream.h>] gain no marks in programming questions. The examiner suspects that the candidates do not understand what purpose they serve in runnable programs.

Examples of particular languages used in the answer points are illustrative only and do not imply that candidates need know these specific languages.

Section A

Question 1

The ATBASH cipher dates back to 500BC and operates by substitution of any letter by the corresponding letter from the opposite end of the alphabet. Thus A → Z, B→Y and so on.

a) Assuming that the letters are ASCII-coded, how would you convert any capital letter to its ATBASH equivalent? Write a suitable data structure to hold every capital letter and its ATBASH equivalent.

(4 marks)

b) Derive a relationship between the capital letters and their ATBASH equivalents. [Marks will be awarded for working as well as the relationship]

(10 marks)

c) Develop the algorithm below which encodes capital letters from a text file <testext> to ATBASH equivalents. Spaces, punctuation are not coded.

(16 marks)

```
Algorithm
Set up Data Structure as defined in part a)
Set up <testext> file for reading
WHILE NOT end-of-file <testext>
        READ <testext>, character
        IF character is a capital letter THEN
                CONVERT character to ATBASH code
                PRINT character, ATBASH code
        ELSE PRINT character
        ENDIF
ENDWHILE
```

Answer Pointers

a)

```
TYPE    At_pair = RECORD
        letter, code_letter : 'A'..'Z'  ( or CHAR)
        place :INTEGER
        END
At_Table : ARRAY[1..26] OF At_pair
```

b)

This relationship is shown in the table below:

Letter	A	B	C	D	E	F	G		U	V	W	X	Y	Z
Position	1	2	3	4	5	6	7		21	22	23	24	25	26
Code letter	Z	Y	X	W	V	U	T		F	E	D	C	B	A
Position	26	25	24	23	22	21	20		6	5	4	3	2	1

The method of conversion is to be:
letter → ORD value → position → code position→ pos→ ORD(pos)→ CHR(ORD(pos))

It can be seen from the table that the relationship between letter(position) and code-letter(position) is code-letter(position) = 26 - letter(position) + 1

letter(position) = ORD(letter) – ASCII_start → apos

The latter term allows for the fact that ORD(letter) is not 1 but a fixed offset more than this, depending on the code the system uses. 26 is the number of letters in the alphabet.
ORD(code-letter) = 26 – apos + ASCII_start
code_letter = CHR(26 – apos + ASCII_start)

c)
Algorithm

development is needed at the points shown ***n**:

```
Set up Data Structure as defined in part a)                    *1
Set up <testext> file for reading                          *2
WHILE NOT end-of-file <testext>
        READ <testext>, character
        IF character is a capital letter THEN
                CONVERT character to ATBASH code   *3
                PRINT character, ATBASH code
        ELSE PRINT character
        ENDIF
ENDWHILE
```

Algorithm Development

```
*1  {starting value}  position = 1
FOR ach := 'A' TO 'Z' DO
        At_Table[position] . letter := ach
        At_Table[position[ . code_letter = _CHR(26 – position + ASCII_start)
        At_Table[position] . place := position
        ADD 1 TO position  {for next cycle}
ENDFOR
*2  ASSIGN FILE <testext> to system file
    RESET(testext)
*3      WHILE NOT EOF(testext) DO
                READ(testext, ach);  WRITE(ach)       {writes code letter}
                apos := 1                                    {start at 1st position}
               IF ach IN ['A'..'Z'] THEN              {IF ach is a letter THEN}
                BEGIN
                WHILE ach <> At_Table[apos].letter DO  apos := apos + 1;
                        code := At_Table[apos].code_letter;
                        WRITE(code);                    {write uncoded letter}
                END
                    ELSE WRITE(ach)                    {write unaltered character}
        ENDWHILE
```

Complete Pascal Program incorporating these results:

{This was not expected in the candidates' answers but is given for those who want to see the conclusion of the problem listed as code}

```
PROGRAM Atbash(INPUT,OUTPUT,textext);
{encodes text from a file <testext> according to the Atbash cypher}
CONST base  = 26;
TYPE
Atpair = RECORD
letter, code_letter:CHAR;
place:INTEGER
END;
VAR testext:TEXT; ach,code:CHAR; ASCII_start,apos,position,ct:INTEGER;
    AT_Table:ARRAY[1..26] OF Atpair;
BEGIN
ASSIGN(testext,'C:\PROPAS\SOURCES\testext');  RESET(testext);
ASCII_start:= ORD('A');  ct := 0; position := 1;
(* set up Atbash equivalents *)
FOR ach := 'A' TO 'Z' DO
   BEGIN
      At_Table[position].letter := ach;
      code := CHR(ASCII_start + base - position);
      At_Table[position].code_letter := code;
      At_table[position].place := position;
      position := position + 1
   END;
WRITELN('Write out letters and ATBASH Equivalents');
FOR apos := 1 TO base DO
   BEGIN
   WRITELN(apos:3,' ',At_Table[apos].letter,' ',
   At_Table[apos].code_letter,' ':3,At_Table[apos].place:2,' ')
   END;
WRITELN;
WRITELN('ENCODE text (cap letters only for simplicity) by ATBASH cypher');
RESET(testext);
WHILE NOT EOF(testext) DO
   BEGIN
   READ(testext,ach);apos := 1;
   IF ach IN ['A'..'Z'] THEN
      BEGIN
      WHILE ach <> At_Table[apos].letter DO apos := apos + 1;
      code := At_Table[ORD(ach) – ORD(‘A’ + 1;
      WRITE(code);
      END
      ELSE WRITE(ach)
   END;
CLOSE(testext,False);
WRITELN
END.
```

Examiner’s Guidance Notes

Very few answers and poorly done by nearly all candidates. Most candidates show little idea of algorithmic development at all. More time should be given in preparatory courses to algorithmic problems of this type, which need not subsequently be developed into runnable programs. In contrast, typing up trivial problems on a PC gives the student the idea that the only difficulty in program writing is getting the syntax right. This is of little value as a preparation for this examination.

Question 2

The simplest Fibonacci series Fib(N) is where each term after the two is the sum of the preceding two as shown in the table below:

Fib(N)	1	1	2	3	5	8	13	21
N	1	2	3	4	5	6	7	8

The following function code is **intended** to return the Fibonacci series term corresponding to the input parameter<N>:

```
line        code
1       INTEGER  FUNCTION  itfib (N : INTEGER)
        VARIABLES   p, term, prev1, prev2 : INTEGER
        BEGIN
2           p  =  3    prev1 =  1    prev2  =  1   term = 1
3           WHILE  p  is LESS THAN  N DO
4               term  =  prev1  +  prev2
5               prev2 =  prev1
6               prev1 =  term
7               ADD 1 TO   p
            ENDWHILE
8           iterfib = term
        END
        ENDFUNCT
```

a) Dry run the function code with N = 5.

(20 marks)

b) Does the variable <term> contain the correct Fibonacci term when the code has executed for N = 5? If not, suggest reasons why not.

(4 marks)

c) The code also contains an error so that a function call such as PRINT itfib(5) receives no return value to print. Explain why and make an appropriate correction.

(3 marks)

d) The code also contains an error relating to N. What is this error?

(3 marks)

Answer Pointers

(a) Dry run the function code with N = 5.

Line no.	Instruct	p	term	prev1	prev2	WHILE	Sundries
1	Function call	?	?	?	?	?	N = 5
2	Assign	3	?	1	1	?	
3	WHILE					true	Loop starts
4	Assign	3	2	1	1		
5	"	3	2	1	1		
6	"	3	2	2	1		
7		4	2	2	1		
3	WHILE					true	Loop continues
4	Assign	4	3	2	1		
5	"	4	3	2	2		
6	"	4	3	3	2		
7	"	5					
3	WHILE	5	3	3	2	false	Loop ends
8	assign function						iterfib = 3

There were 5 marks for choice of column headings and use of line numbers. Other marks were for correctness of the table; a mistake was only penalized once.

b)

<term> does not contain the appropriate value (5).

To achieve this EITHER a starting value of p = 2 OR the condition for the while loop changed to
IF (1 <= N < 3) OR (N <= 0) THEN term = 1
WHILE p LESS THAN OR EQUAL TO N forces another iteration
where term = prev1 + prev2 becomes 3 + 2 = 5.

c)
The assignment is to <iterfib> whereas the function is named <itfib>. We need to change line 8 to itfib = term for a return value to be passed back via the function name. Alternatively, all the names could be changed to 'iterfib'.

d)

The loop condition should be WHILE p LESS THAN OR EQUAL TO N DO…
or WHILE p <= N DO

Examiner's Guidance Notes

Very popular and generally done well. Full range of marks used. About 40% of candidates attempted only part a).

a) Usually correct. The tabular method is universal now. Many candidates correctly drew up the table but didn't use it to help with the answers to subsequent parts. About half of the candidates did the table but didn't attempt the other parts, or got them wrong.

   Candidates should go on to use such tables to spot programming errors, not just memorise the method without any idea why they do it.

b) Those who realized where the fault was usually knew how to correct it.

c) Most spotted the discrepancy between ‘iterfib’ and ‘itfib’ as the cause of error.

d) Only a few got this right. Most made guesses which revealed how little they knew of programming.

Question 3

a) Name, and briefly explain, the operations usually applied to the data structure called a queue.

(5 marks)

b) Write an implementation of a queue (data structure and operations) in a programming language that you know. The values to be stored in the queue will be integers. You may assume that there will never need to be more than 20 values queued at any one time.

(15 marks)

c) Using your implementation as in (b), create a program which processes a queue (MIX) of integers until it is empty, passing even numbers to a second queue (EVEN) and odd numbers to a third queue (ODD).

(10 marks)

Answer Pointers

a)
The names used for queue operations are not universally agreed, but in everyday life we normally speak of **join**ing a queue, being **serve**d in a queue and having some idea of the **length** of a queue, thus a typical answer might be:

isempty() - enquire if the queue is empty,
joinq() - add to the queue,
serveq() - leave the queue

Other vocabulary (e.g. attach, detach) and operations (e.g. initialise) were accepted

b)
The queue data structure contains an array, two pointers indicating the joining and serving points and another value which is the length of the queue. These are conveniently stored all together in a record or a structure. Notice that joinq is a procedure (the job is just done) whereas serveq is a function (returning a result - the element being served). The solution below only contains code for successful operations as this was all that marks were awarded for, but in reality warnings or error messages would have to be generated for trying to join a full queue, serving from an empty queue, etc.

```
typedef struct{
        int member[20];
        int length;
        int joinp;
        int servep;
} queue;

int isempty(queue q){
        return(q.joinp==q.servep);
}
```

```
void joinq(queue q, int e){
        if(q.length<20){ /* there is room in the queue */
                q.member[q.joinp]=e; q.length++;
                q.joinp++; if(q.joinp>=20)q.joinp=0;
        }
}
int serveq(queue q){
        if(q.length>0){ /* there is something in the queue */
                int r=q.member[q.servep]; q.length--;
                q.servep++; if(q.servep>=20)q.servep=0;
                return(r);
        }
}
```

c)

```
queue MIX,EVEN,ODD;

... we do not know how the MIX queue was filled up...

while( ! isempty(MIX) ){
        int e=serveq(MIX);
        if( (e % 2) == 0 )          /* % is the modulus operator, remainder after division */
                joinq(EVEN, e)
        else
                joinq(ODD, e)
}
```

Examiner's Guidance Notes

Not a particularly popular question

a) Many candidates wrote a general description of the way a queue operates and often a diagram of a queue. The question is very specific: name the operations associated with a queue and describe how the operations work.

Push and Pop are usually reserved for a stack.

Note that a definition of serveq which just deletes an element from the stack is almost useless. The operation serveq must return the actual element that has reached the front of the queue, so that as in part (c) something can be done with it (e.g. placing it into another queue.

b) Many answers contained code that was not acceptable for various reasons. The point of defining a data structure with distinctive named operation is that each named operation will become a subroutine (procedure or function). Therefore code to perform joinq or serveq without using a subroutine did not attract many marks. Code to read in data to a queue was not requested and gained no marks.

c) This part was not well answered. The idea was to use the subroutines (procedures or functions) already defined in part (b). However most answers to this part missed the point and wrote out new code from scratch.

Question 4

	program A	program B

	int abs(int x){
	  int a;
	  if(x < 0)
	  a = - x;
	  else
	  a = x;
	  return(a);
	}
	int z = abs("hello");
		FUNCTION abs(x AS integer)
		  VAR a : integer
		  IF x LESS THAN 0
		  a <- - x
		  ELSE
		  a <- x
		  RETURN(a)
		ENDFUNC
		VAR z = abs("hello")

You may elect to answer this question using either program A or program B.

a) Choose either program A or program B and then find and copy out an example of the following:

(1 mark each, total of 15)

i) a reserved word
ii) a variable identifier
iii) an integer constant
iv) a monadic operator (operator with one operand)
v) a type identifier
vi) a diadic operator (operator with two operands)
vii) an assignment symbol
viii) a relational operator
ix) a punctuation symbol
x) a formal parameter
xi) a local variable
xii) a function identifier
xiii) a string constant
xiv) an arithmetic operator
xv) an actual parameter

b) Continuing with either program A or program B as in part (a), find and copy out an example of the following:

(3 marks each, total of 15)

i) a function call
ii) a function declaration
iii) an assignment statement
iv) a variable declaration
v) a conditional statement

Answer Pointers

	program A / Program B
i) a reserved word:	else / ELSE
ii) a variable identifier:	a / a
iii) an integer constant:	0 / 0
iv) a monadic operator:	- / -
v) a type identifier:	int / integer
vi) a diadic operator :	< / LESS THAN
vii) an assignment symbol:	= / ←
viii) a relational operator:	< / LESS THAN
ix) a punctuation symbol:	; / :
x) a formal parameter:	x / x
xi) a local variable:	a / a
xii) a function identifier:	abs / abs
xiii) a string constant:	"hello" / "hello"
xiv) an arithmetic operator:	- / -
xv) an actual parameter:	"hello" / "hello"

b)

i) a function call

```
abs("hello")
```

ii) a function declaration

```
int abs(int x){int a;if(x<0) a=-x else a=x; return(a);}
/
FUNCTION abs(x AS integer)
    VAR a : integer
    IF x LESS THAN 0
        a ← -x
    ELSE
        a ← x
    RETURN(a)
ENDFUNC
```

iii) an assignment statement

```
a = x; / a ← x
```

iv) a variable declaration

```
int a; / VAR a : integer
```

v) a conditional statement

```
if(x<0) a=-x else a=x;
/
IF x LESS THAN 0
    a ← -x
ELSE
    a ← x
```

Examiner's Guidance Notes

A very popular question but not particularly well answered.

a) All the answers were 'one word' answers, i.e. they required copying out one 'word' (or token) from either program A or B. Mostly candidates wrote too much. However some were successful by writing something like "FUNCTION abs(x AS integer)" as an answer to part (xii) where the underlined part was taken as the answer.

Some specific wrong answers were:
i) abs & "hello" are not reserved words, they are programmer defined i.e. invented for this program by the programmer.
iii) x is not an integer constant
iv) abs is a function with one parameter not an operator with one operand
v) var is not a type identifier
vii) ":" is not an assignment symbol
viii) "if" is not a relational operator
ix) The double quotes character does not stand on its own as a punctuation symbol in a programming language - it is always part of a longer word (token) e.g. "hello"
x) & (xv) candidates do not seem to know the difference between a formal parameter (used in a function declaration) and an actual parameter used in a function call
xiv) "<" is not an arithmetic operator because it does not have an arithmetic result - it is in fact a logical or boolean operator

b) In contrast none of these answers were one word answers and many candidates lost marks by writing too little. So, for example, "int abs(int x)" is not the full declaration of function "abs" and "if(x<0)" is only the beginning of an "if" statement.

Section B

Question 5

The time <t> taken by an athlete to complete a lap of a track increases by a constant ratio <f> such that the time for the n^{th} lap is given by t(n) = f * t(n-1).

a) IF the time for the first lap is <inputtime> write a recursive function which works out the time taken for any number of laps<num> given the values <inputtime>, <num> and <f>.

(4 marks)

b) Write a program (or pseudocode) incorporating the recursive function based on the expression derived in part (a). The program requests the time for the first lap <inputtime>, the ratio of the times for successive laps <f> and how many laps <num>. It prints the total time taken for all the laps.

(8 marks)

Answer Pointers

a)

```
PROGRAM runner(INPUT,OUTPUT);
VAR numlaps:INTEGER;  alaptime,ratio:REAL;

FUNCTION time(nlaps:INTEGER; inputime,factor:REAL):REAL;
BEGIN
IF nlaps = 1 THEN time := inputime
      ELSE time := time(nlaps-1,inputime,factor)*factor
END;
```

b)

```
BEGIN (* top level *)
WRITELN('lap times for athlete:Input data items...');
WRITELN('INPUT time for first lap/seconds ');READ(alaptime);
WRITELN('INPUT time ratio for successive laps ');READ(ratio);
WRITELN('INPUT how many laps to be run ');READ(numlaps);
WRITELN('TIME for ',numlaps:2,'  is ',time(numlaps,alaptime,ratio):6:2)
END.
```

Examiner's Guidance Notes

Unpopular, but a few correct solutions. Half marks given for those who wrote a correct iterative function for the lap time calculation; some clearly hadn't covered recursion. Most functions had only one parameter, not enough in this case. Some copied out the recursive function for 'factorial(n)' here. Some very poor answers didn't even input the necessary data for the calculation, for which there were quite generous marks. Candidates who haven't studied recursion should leave this question alone.

Question 6

a) An inter date is given in the form ddmmyy where dd = day digits, mm = month digits, yy = last two digits of the year. Thus 251207 is 25th December 2007. Write code or pseudocode to re-arrange the date into the form yyymmdd using the MOD and DIV operators where MOD gives the modulus and DIV is the integer division operator; e.g. 10 MOD 3 = 1;10DIV 3 = 3

(4 marks)

b) Write code to sort a sequence of these modified dates held in the array LONGDAT[1..99] into descending order of year.

(8 marks)

Answer Pointers

a)

```
yeardigits = date MOD 100
daydigits := date DIV 10000
monthdigits  = (date - 10000*daydigits) DIV  100
newdate  =  (2000 + yeardigits) * 10000 + 100 * monthdigits + daydigits
```

b)

```
longdat:ARRAY[1.99] OF items
PROCEDURE swap (VAR data1,data2:INTEGER);
VAR temp:INTEGER;
BEGIN
temp := data1;
data1 := data2;
data2 := temp
END;  {swap code}

BEGIN {top level}
{input data items} [already held in array <longdat>]
{sort}
FOR outer := 1 TO (nitems -  1) DO
        FOR  inner := (outer + 1)  TO nitems DO
                IF longdat[outer] < longdat[inner]  THEN swap(longdat[outer],longdat[inner]);
END;
{other sorting methods were allowed here}
```

Examiner's Guidance Notes

Moderately popular as the sort part (b) is nearly always covered as bookwork. This was usually written out in the form candidates had memorized. Some strange ideas on the use of MOD and DIV operators in part (a), especially from those who used character input!

Question 7

a) Write a suitable data structure to hold a node of a linked list with one pointer and 5 data items relevant to the stock of a garden centre. These are:

name (20 characters)
plant_type (indoor OR outdoor)
season (one from spring, summer, autumn, winter)
price (pounds and pence)
date stock received
State which language you are using.

(4 marks)

b) Write a procedure or function called 'onend' to add a new item at the end of the list, opposite to the head. Assume the parameters for "onend" consist of the head pointer, and the item to be added to the end.

(8 marks)

Answer Pointers

a)

```
TYPE   aptr = ^ node;
       node = RECORD;
       name : PACKED ARRAY [1..20] OF CHAR;
       season : (spring, summer, autumn, winter) { enumerated type}
       atype : BOOLEAN  or (indoor, outdoor)
       price : REAL;
       date_rec : date OR long_INTEGER;
       next : aptr
         END;
```

Alternatives were allowed for those who didn't know about enumerated type. One mark was given for realizing a record was needed; another 1or 2 marks for the pointer fields.

b)

```
PROCEDURE onend (VAR head : aptr; item : aptr);
VAR
BEGIN
NEW(newptr);
newptr.title := aptr.title;
newptr.price := aptr.price;                  {some systems allow newptr := aptr }
newptr. numsold := aptr.numsold
temp := head;
prev := head;
{cope with empty list}
IF head = NIL THEN head := newptr
       ELSE BEGIN   {search for end of list}
       temp := temp^.next;
       WHILE  (temp <> NIL)  DO
                                     BEGIN
                                     temp := temp^.next;
                                     prev := temp
                                     END
       END
newptr := prev;
newptr.next := NIL;
ENDPROC
```

Examiner's Guidance Notes

Quite popular. Those who learned Basic as their programming language should have left this question alone. Many candidates attempted only part (a) the record declaration. Some wrote extra procedures needed in a program actually to be implemented (like data input) but which was not asked for in the question. Many answers to (b) had only a lengthy interactive input sequence. Weaker candidates did not realize that list traversal using a loop was necessary part of the procedure. A linked list question does attract 'brain-dump' answers where the student delivers what they have memorized about linked lists hoping that some of it answers the question.

Question 8

a) Letters to candidates containing the results of intermediate examinations were produced by a PC system taking its data from a candidates results file <results> and a name file <names>. Draw and label a schematic diagram showing what devices/components are needed to produce these letters.

(6 marks)

b) For failed candidates, the program produces the sentence

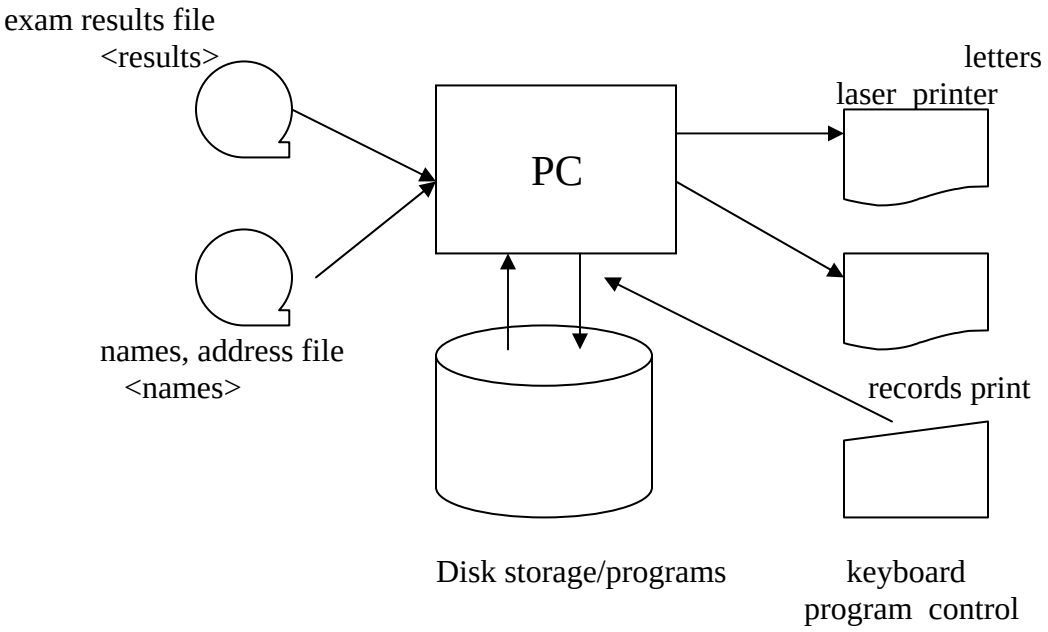
You have FAILED in the following subjects….

This must now be altered to *You have NOT YET PASSED the following subjects...* What system documentation should be available in order to under take this alteration?

(6 marks)

Answer Pointers

a)



Entirely written answers were only penalized 1 mark if they used description rather than produce a diagram, as required in the question.

b)

The system documentation MUST contain (a) details of the program environment, how to load and run the program, system requirements. (b)Annotated source listing if the program is long or not well commented. Thus it will be possible to locate how the message '*sentence You have FAILED in the following subjects…*' is stored. As the replacement message is longer, you will need to know if its length is stored by the system or not. Finally, it will be necessary to re-compile the amended program and test it with appropriate data before running the system with live addresses.

Examiner's Guidance Notes

This question attracted many poor-quality answers. Significant number of answers had screen designs for Visual Basic implementation but had no idea how to answer (b) which involved documentation! Others mentioned altering source code without mentioning the importance of technical manuals, even facilities for that implementation. Few mentioned that the amended source code may need recompiling; not always easy if the programming language is old and compilers are hard to come by for that version.

Question 9

a) What is normally meant by the term 'debugging'?

(4 marks)

b) How is debugging approached in a simple programming environment where the programmer has only the standard output facilities of the programming language to use for this purpose?

(4 marks)

c) What extra facilities to assist in debugging might be provided in a more extensive development environment?

(4 marks)

Answer Pointers

a) The program does not respond in the way that was expected and the programmer is looking for more diagnostic information to discover the source of the problem and fix it. Debugging is the stage after the syntax is correct.

b) The programmer can insert extra output statements in key places in the code. It is usual to output a distinctive label to show that this place has been reached at run-time and often it is useful to output the value of some key variables. Key places in the code are just inside loops (to find out how many times they are repeating) and on the separate branches of conditionals (to understand which branch was taken).

c) In a more sophisticated development environment the programmer might expect to be able to set 'breakpoints' where the program can be temporarily suspended and inspections made of the values of variables, etc.

Examiner's Guidance Notes

a) The essence of debugging is to find and correct errors. There was confusion between testing and debugging. If you test a program, you may find that it has an error (i.e. you know there is an error somewhere - perhaps because the program gives some wrong output). Debugging is the business of finding the exact point in the code which needs to be changed and making the change to put it right.

The word debugging is NOT normally used for the process of removing syntax errors from code which prevent the code from compiling. Debugging is reserved for programs that are running, but not successfully - i.e. dealing with the more serious logical errors.

b) Candidates wanted to talk about black box testing and white-box testing. However, black box testing cannot be called debugging because the source code is not available. Even white-box testing is not debugging (although the source is available) for the reason given above which distinguishes testing and debugging. Debugging is the process that follows on after testing has shown up a problem.

c) Candidates wrote about systems which 'automatically' correct errors.

Some candidates wrote about bringing in a compiler or an interpreter at this point, but one or other of these tools must have already been used in part (b).

Question 10

A new electronic device that stores music and connects to the domestic television is being developed. It is to be controlled solely by a remote control which has only 5 buttons labelled *up*, *down*, *left*, *right*, *select*.

a) Consider each of the five main interactive screen interface elements found on web pages (and elsewhere) and describe which elements can be used easily with this restricted remote control.

(8 marks)

b) Describe one way of implementing textual input using this interface.

(4 marks)

Answer Pointers

a) push button - can use up, down, left, right to navigate to button, select to push
radio button - can use up, down, left, right to navigate to button, select to push
check box - can use up, down, left, right to navigate to button, select to push
drop-down menu - can use left right to navigate to menu;
up, down to navigate to chosen item; select to select item
text box - cannot be used

b) To simulate a text box, show a display of characters (possibly, but not necessarily in traditional keyboard layout) which work as buttons. Use up, down, left, right to navigate to a character, select to select character. Repeat to enter rest of text. Need an OK/Done button to signal end of text entry.

Examiner's Guidance Notes

a) Half of the marks were awarded for mentioning each of the four main interface elements that could be used easily (1 mark each). The other half of the marks was reserved for giving some way in which the interface elements could be operated by the remote control.

Some students picked other parts of the interface e.g. labels. Although labels are an important, even essential part of the interface, they are not among the INTERACTIVE elements in the interface.

b) Many candidates suggested using a text box and had thus clearly missed the point of the question. Other answers were spoilt by the unacceptable use of the word "alphabet" in place of "character". So for example an answer might contain the phrase "the user would see a display with 26 alphabets" when it should be "the user would see a display with 26 characters". [An alphabet is a specific collection of characters - e.g. the English alphabet, the Chinese alphabet.]

It was curious that some candidates could give a good answer to (b) and gave either no answer or a poor answer to (a).

Question 11

A prospective buyer has come into a computer shop to discuss which computer to buy on the basis of the software provided. The buyer has started asking questions about virus checking, maintaining an address book, doing backups and storing digital photos.

As the shop assistant, what would your advice be to the customer? For each of these four areas you should state whether you consider system or application software to be applicable and what features of the software you would consider important.

(12 marks)

Answer Pointers

i) virus checking - system software
important features - e.g. regular updates, easily configured schedule
ii) addressbook - application software
important features - e.g. integration with other systems (e.g. email), import, photo, groups
iii) backup - system software
important features - e.g. incremental backup, automated by time of day/week
iv) photos - application software
important features - e.g. store in albums, add titles, search by title/tag, editing/retouching

Examiner's Guidance Notes

Many answers were badly laid out - e.g. a full page of writing with no headings, paragraphs, underlining, etc.

This was NOT a question about recommending a particular computer or operating system. The question is not about hardware - only about software.

The question did NOT ask what is a virus checker, what is an addressbook, etc. It did not ask how they work.

There was some confusion caused by candidates reading "maintaining an addressbook" as "maintaining and addressbook", so the answer gave separate advice about computer maintenance and keeping an addressbook.

The question did NOT ask for a definition of system software and application software so the effort of writing these into answers was wasted.

There was confusion over system software and application software. It is not correct to assume that everything delivered with the computer initially is system software.

In addition to specific features of particular software, other more general features were accepted like ease of use, reliability, efficiency, cheapness, value, etc. However, each of these general features was only allowed once each.

Question 12

Because of a company takeover a new team of software engineers have taken over a package that had been written by the previous team. Amongst the team there is

i) a group charged with running the package for the next few months
ii) a group who are looking at recoding the package line-by-line from the existing programming language to the company's preferred coding language
iii) a third group looking at rewriting the package from scratch.

Explain with reasons what kind of documentation each group would be looking for in order to do their respective tasks.

(12 marks)

Answer Pointers

a) running package - need user's guide - detailed guide on how to load and run the package with all the options available
[other answers offered Review & Maintenance, User's Documentation, User Manual, Maintenance Documentation]

b) recoding - need functional design - detailed guide to all the routines with function interfaces, parameter descriptions, etc - HOW has the package been written
[other answers: Programmer Documentation, Technical Documentation, System Design, System Documentation]

c) rewriting - need requirements analysis / specification - what does the package need to do - WHAT is in the package
[other answers: Requirements Specification, Prototype]

Examiner's Guidance Notes

This question was intended to explore the documentation required at different stages of the Software Development Life Cycle.

The question was quite well answered by candidates who understood the question, although a wide range of names were given to the various pieces of documentation - hence the importance of explaining what is meant by each type of documentation.

It was clear that in some cases the word 'scratch' was not understood or misunderstood - one candidate thought it was a new programming language! Also the word 'charged' may have been

misunderstood leading to answers discussing how much it was going to cost to recode the package and whether it was going to be worthwhile.

It was expected that each group would require different documentation and so answers which gave the same documentation to different groups were penalised.

Several candidates insisted on discussing what documentation the groups needed to WRITE, when the question was what did they need to READ. Other similar misunderstandings showed because candidates wrote about groups having to do things that were not intended by the question. Group (i) do not have to do any testing of the existing package. Group (ii) do not have to do any white box testing of the existing package. Group (iii) do not have to carry out a requirements analysis or a cost analysis.

Note the distinction in this question between 'language' and 'code':
The group needs to know the existing language - e.g. C Java; i.e. a one word answer - the name of a programming language.
The group needs to know the existing code - e.g. they need to be handed a DVD containing all 10,000 lines of the existing code.