BCS HIGHER EDUCATION QUALIFICIATIONS
BCS Level 4 Certificate in IT

April 2010

EXAMINERS' REPORT

SOFTWARE DEVELOPMENT

General Comments.

The standard of the programming work continues to be very low. Few candidates could do more than input a few numbers and (say) print the average.

There were much better answers to descriptive questions concerning the Web. Many candidates rely on memorized material alone to pass this paper, and obviously make very little preparation for the programming area. Some of the descriptive answers ran to FOUR pages.

Annoyances were:- Not filling in the 'Questions Answered' box (30%), illegible question numbers (10%), omission or writing of the wrong question numbers on the front cover.

Question A1

A1 The volume of a cylinder with outer radius (r_1), the inner radius (r_2) and height (h) is given by the formula:

$$volume = \pi*(r_1^2 - r_2^2)*h$$

The weight of a cylinder depends on the density of the material from which it is made:

$$weight = density * volume$$

Cylinders are to be made from melted-down silver coins, each weighing 3.9 grams.

$$Density\ of\ silver = 10.5\ g\ cm^{-3}$$

A program is to print a table of the various sizes of cylinder possible by varying r_1, r_2 and h all between appropriate input limits. Also the weight is to have a maximum value (*maxwt*).

a) Sketch the envisaged cylinder and state what limits must be applied to the calculations of its size and weight.

(4 marks)

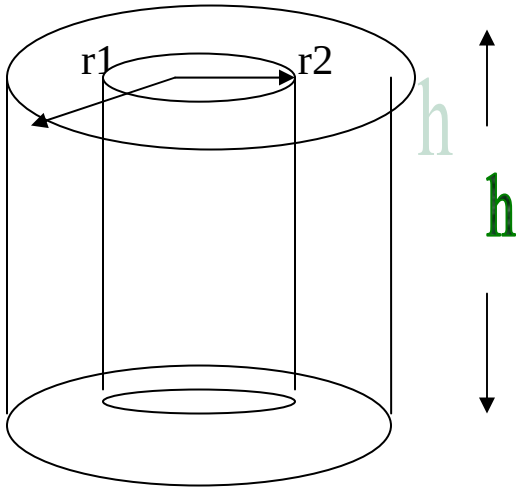
b) Develop an **algorithm** to print a table of the various sizes of cylinder that can be made and how many coins are required. Each cylinder must not weigh more than 1 kilogram or be greater than 10 cm in height. The cylinder dimensions are to be input as maximum and minimum values for both (r_1) and (r_2). The increment between successive calculations (incr) is also input.

The algorithm must have at least two stages of development and need not be developed beyond the point at which coding / testing would be straightforward.

(12 marks algorithm)
(14 marks development)

Answer pointers

a)



The diagram shows that the items to be varied are:-
The height (h) of the cylinder,
The outer axis of the circle (r1)
The inner axis of the circle (r2)

b) Develop an algorithm to print a table of the various sizes of cylinder that can be made and how many coins are required

The weight of the cylinder is dependent on (h, r1, r2), related by
Volume = $\pi*[(r1)^2 – (r2)^2]*h$
Weight = density * volume = $10.5* \pi*[(r1)^2 – (r2)^2]*h$

So the initial algorithm could be

```
INPUT cylinder data [h, r1, r2] with prompts
INPUT maxwt
FOR h = hmax to hmin
  FOR r1 = amax to amin
    FOR r2 = bmax to bmin
        Calculate cylinder weight
```

```
        numcoins = cylinder weight /  density
        IF weight is below the maximum THEN PRINT radii, weight
    ENDFOR
  ENDFOR
ENDFOR
```

It is apparent that a 'FOR' loop runs through all the values of the control variables, even when successive weights are above the maximum. Consider instead 'WHILE' loops which can have an additional condition to stop further calculation when the weight is above the maximum.

```
INPUT cylinder data (amax, amin, bmax, bmin, hmax, hmin, incr, maxwt)
h = hmin
WHILE h < = hmax
        Set initial a value
        WHILE  r1 < amax
                Set initial r1 value
                WHILE r2 < bmax
                        CALCULATE area = outer area – inner area
                        CALCULATE weight of cylinder
                        IF weight < maxwt  THEN
                        BEGIN
                                CALCULATE number of coins = weight of cylinder / weight
                                of single silver coin
                                PRINT area, height, a, b, weight, number of coins
                        ENDIF
                         r2 = r2 + incr
                ENDWHILE
                r1 = r1 + incr
        ENDWHILE
        h = h + incr
ENDWHILE
```

(12 marks)

Development.

Area of metal = (outer circle area) – (inner circle area) = π*(r1*r1 – r2*r2)

Appropriate formatting of the output should also be considered. Obviously this depends strongly on the target language.

A table caption should be printed outside any of the loops.

Examiner's Guidance Notes

Very poor quality answers. Very few candidates understood what 'Algorithmic development' was, let alone apply it to the problem. At least 10% of candidates wasted time by copying out the question. Over 90% did not attempt any development. It is obvious that nearly all of candidates have not progressed beyond the kind of problem which reads in a few numbers and prints the average.

Candidates must be taught algorithmic development if they are to attempt 'A' section programming questions.

Question A2

A2 a) Define recursion.

(5 marks)

b) The following definition of function *sub* uses recursion. Dry run the call of *sub(a,b)* with values *a* = 5 and *b* = 2 using either version A or version B. All the variables contain integer values.
(20 marks)

line	Version A	Version B
1	FUNCTION sub(a,b)	int sub(int a, int b){
2	IF b > 0 THEN	if(b > 0)
3	sub := sub(a-1, b-1)	return(sub(a-1, b-1));
4	ELSE	else
5	IF b = 0 THEN sub := a	if(b == 0) return(a);
6	ELSE	else
7	sub := sub(a+1, b+1)	return(sub(a+1, b+1));
8	END	}

c) Which line in the function code is not tested in this dry run?
Give, with reasons, different values for *a* and *b* which would cause this line to be tested. Do NOT carry out another dry run.

(5 marks)

Answer pointers

a)

Functions are recursive if they call themselves. They must include a terminating clause

b) {dry run}

line	instr	a	b	IF	Output/notes
1	FUNCTION	5	2		
2	IF			TRUE	b > 0
3	Sub call				
1	FUNCTION	4	1		
2	IF			TRUE	
3	Sub call				b > 0
1	FUNCTION	3	0		
2	IF			FALSE	b = 0
4	ELSE				continue
5	IF			TRUE	RETURN sub = 3
8	END				

c) Line 7 was not called as ***b*** was always greater than ***a*** until it became zero.
If we instead had ***a*** = -5 ***b*** = -2 then line 7 will be repeatedly called until ***b*** becomes zero.

Examiner's Guidance Notes

Some good answers. Candidates are obviously prepared for this type of question.

a) Some weird ideas about recursion but majority have learnt a definition

b) Generally done well despite some careless errors. Many answers spoilt by crossing-out and altered figures. Candidates should learn how to carry out a dry run without either of these annoyances.

c) Done well if the dry run was OK.

Question A3

A3 The array ***z*** has been initialised as follows

index	0	1	2	3	4	5	6	7	8	9
z	42	19	55	90	81	76	12	49	0	0

The function ***u*** in the code below is going to be executed with parameter ***v*** set to 7 and parameter ***w*** set to 70. [*You can choose to follow either version of the code*]

a) Trace the call of the function ***u(7,70)*** and state the value that is returned from the call.

(8 marks)

	Version A	Version B
1	int u(int v, int w){	FUNCTION u(v, w : INTEGER):INTEGER;
2	int x, y;	VAR x, y : INTEGER;
3	/* begin function */	BEGIN
4	y = 0;	y := 0;
5	for(x=0; x<=v; x++)	FOR x := 0 TO v DO
6	{	BEGIN
7	if(z[x] > w) y++;	IF z[x] > w THEN y := y + 1;
8	}	END;
9	return(y);	u := y
10	}	END;

b) Write a brief summary of what the function does.

(6 marks)

c) Decide on better names for the identifiers (the function name, its parameters and the variables) and rewrite the code [either version A or version B] using your new names and including suitable comments.

(10 marks)

d) Rewrite lines 5 to 8 [of either version A or version B] using a while-loop instead of a for-loop.

(6 marks)

Answer pointers

a)

x	z[x]	z[x]>70	y
0	42	42>70 = false	0
1	19	19>70 = false	0
2	55	55>70 = false	0
3	90	55>70 = true	1
4	81	81>70 = true	2
5	76	76>70 = true	3
6	12	12>70 = false	3
7	49	49>70 = false	3

b) The function u searches the array z between indexes 0 and v (7) and counts how many elements have a value greater than w (70).

c) Looking for names that suggest the meaning of the variables in the program

u -> countGT,
v -> upper
w -> min
x -> index
y -> count

Rewriting the code with the new (meaningful) names

	Version A	Version B
1	int countGt(int upper, int min)	FUNCTION countGt(upper, min : INTEGER):INTEGER;
2	{	VAR index, count : INTEGER;
3	int index, count;	BEGIN
4	count = 0;	count := 0;
5	for(index=0; index<=upper;	FOR index := 0 TO upper DO
6	index ++)	BEGIN
7	{	IF z[index] > min THEN count :=
8	if(z[index] > min) count ++;	count + 1;
9	}	END;
10	return(count);	u := count
	}	END;

Example comments:
line 1: declare function to count the number of values in array up to element upper that are greater than min
line 2: index for iterating through the array, count for holding the count of large numbers
line 5: loop to visit all the elements of the array from 0 to upper
line 7: if the array at element index is big enough, increment count
line 9: return the final count as the result of the function

d) Rewriting lines 5-8 using a while loop

	Version A	Version B

5	index = 0;
	while(index<=upper){
	if(z[index] > min) count ++;
	index ++;
	}
		index := 0;
		WHILE index < upper DO BEGIN
		IF z[index] > min THEN count := count + 1;
		index := index + 1
		END

Examiner's Guidance Notes

This was a popular question

a) Getting the 'right answer' is only worth half the marks. It was puzzling to find candidates answering parts (b), (c), (d) while leaving part (a) blank. A few candidates had for example z[42], z[19] in their traces, showing a confusion between the subscript/index of an array and the contents at that subscript/index.

b) Precision was required. There are six points to be made. "The function u searches the **array z** between indexes **0** and **v** (7) and **counts** how many elements have a value **greater** than **w** (70)."

The function does NOT sum the values in the array z.
The function does NOT find the maximum value in the array z.
The function does NOT find the sum of the values in z that are bigger than 70

c) When choosing new names some candidates only gave a new name for the function, or for the function and parameters, but not local variables. Some of the new names given were no better than the original (e.g. param1, param2, var1, var2, temp1, temp2). Some candidates named the parameters in pairs e.g. min, max or start, end. These were not accepted as it showed a misunderstanding of the purpose of the parameters.

Comments were often several paragraphs of separated text. What was required was in-program comments. Marks were not given for superficial comments like 'start of function', 'start of loop', 'end of loop', 'end of function'.

d) This part was done quite badly. Some candidates merely changed the word 'for' to 'while', hoping that would be enough - it was not. Many others put a reasonable condition after the while, but then forgot to add in the counting code (index=0; and index++;). Those that included the counting often put it in the wrong places (e.g. index=0 inside the loop, or index++ outside the loop)

Question A4.

A4

program A	program B
/* PROGRAM A */ char near; float sq(float p){return(p*p);} float dist(float x1, float y1, float x2,	PROGRAM B; VAR near : CHAR; FUNCTION sq(p:REAL):REAL;BEGIN sq:= p*p END;

```
float y2){
   float res;
   /* begin function */
   res = sqrt( sq(y2-y1) + sq(x2-x1) );
   return(res);
}
void main(){
   if( dist(8.0, 4.5, 9.0, 3.3) < 10 )
      near = 'Y';
   else
      near = 'N';
}
```

```
FUNCTION dist(x1,y1,x2,y2:REAL):REAL;
   VAR res : REAL;
   BEGIN
      res := sqrt( sq(y2-y1) + sq(x2-x1) );
      dist := res;
   END;
BEGIN {main program}
   IF dist(8.0, 4.5, 9.0, 3.3) < 10 THEN
      near := 'Y'
   ELSE
      near := 'N';
END.
```

a) Choose either program A or program B and then find and copy out an example of each of the following.
[*Take care to copy out only what is requested, nothing more*]

(1 mark each, 12 total)

a.1) a reserved word	a.7) an arithmetic operator
a.2) a variable identifier	a.8) a relational operator
a.3) a type identifier	a.9) a formal parameter
a.4) a function identifier	a.10) an actual parameter
a.5) a character constant	a.11) a local variable
a.6) an integer constant	a.12) an assignment symbol

b) Continuing with either program A or program B as in part (a), find and copy out an example of each of the following.
[*Take care to copy out only what is requested, nothing more*]

(3 marks each, 18 total)

b.1) a function call	b.4) an assignment statement
b.2) a function declaration	b.5) an expression with a boolean (logical) value
b.3) a variable declaration	b.6) a conditional statement

Answer pointers

	program A	program B
a.1	if, else, return	PROGRAM, FUNCTION, BEGIN, END, IF, THEN, ELSE, VAR
a.2	res, near	res, near
a.3	float, char, void	REAL, CHAR
a.4	dist, sqrt, sq	dist, sqrt, sq
a.5	'Y', 'N'	'Y', 'N'
a.6	10	10
a.7	+, -, *	+, - *
a.8	<	<
a.9	p, x1, y1, x2, y2	p, x1, y1, x2, y2
a .10	8.0, 4.5, 9.0, 3.3, x2-x1, y2-	8.0, 4.5, 9.0, 3.3, x2-x1, y2-y1
a.11	y1	res
a.12	res	:=
	=	

b.1	sq(y2-y1) dist(8.0,4.5,9.0,3.3)	sq(y2-y1) dist(8.0,4.5,9.0,3.3)
b.2	float sq(float p){return(p*p);}	FUNCTION sq(p:REAL):REAL;BEGIN sq:= p*p END;
b.3	char near; (or) float res;	VAR near : CHAR; (or) VAR res: REAL;
b.4	near = 'Y'; (or) near = 'N'; (or) res = sqrt(…)	near := 'Y'; (or) near := 'N' (or) dist := res (or) sq := p*p
b.5	dist(8.0,4.5,9.0,3.3) < 10	dist(8.0,4.5,9.0,3.3) < 10

b.6	if(dist(8.0,4.5,9.0,3.3) < 10)
	near = 'Y';
	else
	near = 'N';
		IF dist(8.0,4.5,9.0,3.3) < 10 THEN
		near := 'Y'
		ELSE
		near := 'N';

Examiner's Guidance Notes

This was a popular question, but not necessarily answered very well. There were many cases of apparently confident answers to part (a) and all of them wrong!

a) The question is very specific about what is to be copied out, therefore it is wrong to write more than is required unless the required part is underlined or otherwise highlighted. Therefore for part (a) [copy out a reserved word] an answer that reads

if (dist(8.0, 4.5, 9.0, 3.3) < 10)

would be marked incorrect whereas

if (dist(8.0, 4.5, 9.0, 3.3) < 10)

would be marked correct. But the actual answer required was

if

Sometimes there are two or more valid answers for a particular part. If a candidate gave two or more alternative (correct) answers then this was accepted (but did not score extra).

Some candidates said that there was no integer constant, some gave examples like 4.5 which is a 'real' or 'float' constant (i.e. a fractional constant), not an integer.

It was puzzling to see punctuation like ',' and ';' being offered as examples of an assignment symbol.

return(…) is NOT a function call.

b) Some students made up pieces of program to suit the question parts rather than copy out extracts from the given program. This was not accepted.

It seemed that many students were guessing at the answers, often writing down the same thing for several parts, presumably in the hope that one of them might be right.

Unfortunately many candidates got parts b.1 and b.2 the wrong way round.

The answer near = 'Y'; was often offered (wrongly) as an expression with a boolean (logical) value (b.5). This I suspect is the result of pronouncing it as "near **equals** Y" when it should be pronounced "near **is assigned** Y".

SECTION B

Question B5

B5 Write a program to input a value *limit* and print a table of all the odd numbers *n* between 3 and *limit* where (n^2 – 1) is divisible by 8.

(12 marks)

Answer pointers

Table of integers 3 < n < limit where (n^2 -1) is divisible by 8.

```
PROGRAM numdiv8(INPUT,OUTPUT);
VAR limit,n,num,val:INTEGER;
BEGIN
        WRITELN('INPUT UPPER LIMIT FOR TESTS'); READ(limit);
        WRITELN('--n----nsq-1--MOD 8 ----result');
        FOR n := 3 TO limit DO
        BEGIN
                IF (ODD(n)) THEN
                BEGIN
                        val := n*n - 1;
                        num := val MOD 8;
                        WRITE(n:3,' ':3,val:4,' ':2,num:4,' ':6);
                        IF num = 0 THEN WRITELN(' divisible')
                                ELSE WRITELN(' NOT divisible');
                END
        END
END.
```

Examiner's Guidance Notes

Only a few answers and rarely done well. This question required some algorithmic development which was beyond most candidates.

Question B6

B6 Data associated with each of the chemical elements is as follows:

Element name	Up to 12 characters
Symbol	one or two letters
Metal/ non-metal	Boolean
Atomic number	integer
Atomic mass	real number
Density	real number

Melting point	real number
Boiling point	real number

(a) A serial file **atomdata** contains records for all the elements. Write a record description named **elements** for this data and file declaration.

(3 marks)

(b) Write a program which searches the file for the first five metals with a density greater than 8.0 $g.cm^{-3}$ and a liquid range (Boiling point - Melting point) greater than 500^{o}C. Print the name, and atomic mass of the five elements.

(9 marks)

Answer pointers

a)

```
Elements = RECORD
        symbol : ARRAY[1..2] OF CHAR;
        metal/non metal : BOOLEAN;
        atomic_number :INTEGER;
        atomic mass,mpt, bpt : REAL
END;
atomdata : FILE OF elements
```

b)

```
OPEN atomdata FOR READING
Searching = TRUE
count = 0
WHILE NOT EOF(atomdata) AND searching DO
        READ(atomdata, oneatom)
        WITH onatom DO
                liqrange = Bpt – Mpt
                IF density GREATER THAN 8.0 AND liqrange GREATER THAN 500
THEN
                        FOR pos = 1 TO 12 PRINT   name[pos];
                        PRINT symbol[1],symbol[2]
                        PRINT atomic_mass
`                       PRINT density
                        IF metal_nonmetal THEN PRINT ‘metal’ ELSE PRINT ‘non-metal’
                        PRINT ‘liquid range’, liqrange
                        ADD 1 TO count
                ENDIF
                IF count >= 5 THEN searching = FALSE
ENDWHILE
ENDPROG
```

Examiner’s Guidance Notes

Fairly popular question. Most only attempted the record description in part (a). Some wasted time by copying it out again in part (b). Reasonable answers to part (b) from those who had studied serial files.

Question B7

B7 Values for the sine function are obtained from the power series

$$\text{Sin}(x) = x - x^3 / \text{fac}(3) + x^5 / \text{fac}(5) - x^7 / \text{fac}(7) + \ldots$$

where fac(n) = factorial n = 1*2*3*4…n

(a) Write code for fac(n); any method may be used.

(4 marks)

(b) Incorporate your function into another function asine(x) which calculates sine(x) using the power series given earlier. Show how to terminate the calculation when the difference between successive terms is less than 0.00005.

(8 marks)

Answer pointers

a)

```
FUNTION fac (n:INTEGER):INTEGER;
    IF n = 0 THEN fac = 1
    ELSE fac = n*fac(n – 1)
ENDFUNC
```

b)

```
FUNCTION asine(x)
    accuracy = 0.00005
    sum = 0
    mult = x
    previous term = x
    diff = 1
    WHILE diff IS GREATER THAN accuracy DO
        mult = mult * x * x
        term = mult / fac(n + 2)
        sum = sum + term
        diff = abs(previous term – term)
        previous term = term
    ENDWHILE
    asine = sum
ENDFUNC
```

Examiner's Guidance Notes

Unpopular and poorly done question. Many attempted only part a).

Question B8

B8 Write a program to merge two distinct lists into a single linked list. Both of the initial lists contain integers and both are in ascending integer order. This order must be preserved in the merged list. The two lists are of the same length. They are set up by the procedure **setuplist(head)** which returns a pointer to the head of the list. *You must not write code for the procedure setuplist but indicate where appropriate calls to it would be made in your code*

(12 marks)

Answer pointers

Algorithm / pseudocode

```
Variable Declarations
Code for setuplist(head)
BEGIN {top level}
        Setuplist(head1)
        Setuplist(head2)
        first ← head1;   second ← head2;
        head3 ← NIL;
        WHILE ((first <> NIL) AND (second <> NIL)) DO
        BEGIN
                NEW(third);
                 third^.next ← head3;
                IF first list data item >= second list data item THEN
                        Copy FIRST list member into new list
                ELSE
                        Copy SECOND list member into new list
        head3 ← third
        END
END
```

Examiner's Guidance Notes

The least popular question. Of those who did, very few had any idea how to tackle it. Marks were awarded for diagrams of pointer movements, although these were not asked for. This area of the syllabus seems to be ignored by most providers.

Question B9

B9 One of the common operations required of a computer program is to search for a specific item in a collection of items. Describe (either in words, or by pseudo code or by actual program code) one efficient algorithm for searching, taking care to explain what arrangements should be made to make the search efficient.

(12 marks)

Answer pointers

Expect binary chop (or maybe HASH table)
Binary chop

1. information must be ordered according to some key
2. imagine the keys in ascending order in an array
3. to find key k
4. look at the value stored half way down the array, compare with k
5. if equal to k, search succeeds, stop
6. if k is smaller throw away top half of array, otherwise throw away bottom half
7. repeat from step 4 until only one key left

Examiner's Guidance Notes

Very popular, though not necessarily well done. The most common comment on the marked script was "No algorithm".

The sequence

- declare item
- read into item
- search for item
- output result of search

does not constitute a search algorithm

Only half marks were available for a linear search because that is not an efficient algorithm (as requested in question)

The question asked for an algorithm for searching (not sorting) so there were no marks e.g. for the bubblesort algorithm.

There were no marks for discussing what should be typed into a Google search box

Among those who gave an attempt at an algorithm there was confusion between an array index and the array value at that index. For example working in the array v from indexes start to stop, several said "set the variable mid to (start+stop)/2". That is fine. But then they would go on to say "if mid=item …" when it should have been "if v[mid]=item…".

Question B10

B10 Compare the following pairs of terms. [*You are advised that three well chosen sentences per pair will be sufficient – one sentence describing the first term, one sentence describing the second term and a final sentence highlighting the difference between the terms.*]

a) A compiler and an interpreter

b) Internal and external memory

c) Design and implementation

d) System software and application software

(12 marks)

Answer pointers

a)
Compiler: translates high level code to low level code as a complete task
Interpreter: translate high level code to low level code a section at a time
Difference: compiler equivalent to task of translate a book into a foreign language, interpreter equivalent to task of translating a live speech, sentence by sentence

b)
Internal memory: memory directly accessible by the CPU
External memory: memory indirectly accessible, removable (easily)
Difference: internal smaller, faster, direct access – external larger, slower, serial or direct access

c)
Design: Choose data structures, choose algorithm, create pseudo code, may involve ER diagrams, dataflow diagrams, database design, flowcharts.
Implementation: Choose the programming language and, using the design ideas, write the final code.
Difference: design is characterized by diagrams on paper, implementation by executable code.

d)
System software: designed to offer easy access to features provided by hardware (e.g. file access by name, provides all the hardware addressing and load, save)
Application software: designed to enable user to accomplish common user tasks (e.g. in word processing provides WYSIWYG editing)
Difference: does it interface directly with hardware or does it deal directly with user requests

Examiner's Guidance Notes

A popular question. Often the quality of the 4 parts was very uneven

a) Lots of spelling mistakes for compiler!
Some candidates seem to think that a compiler or an interpreter is only for detecting errors.
It is not possible to say that a compiler is faster than an interpreter (or vice versa). A compiler process code that executes faster, but someone using an interpreter will usually get to the point of executing an instruction quicker than someone using a compiler.

b) Candidates made observations about the speed, the capacity, the volatility and the cost of different memory. These were accepted if sensible.

c) This part produced the weakest answers.

It was not enough to locate the phase in the SDLC (this comes before that, etc), some descriptive words about what goes on in the phase were required
Some candidates described implementation as taking the finished code back to the customer and installing it in its intended environment. This was acceptable.

d) A recurring mistake in this part was to say that "Application Software was required to **run** an application"

Question B11

B11 A business selling to the general public via a web site will require that site to have good user-interface design.

a) What are the basic elements that appear on the page with which the user can interact?

(6 marks)

b) Briefly describe what you consider to be the basic concepts of good user-interface design?

(6 marks)

Answer pointers

a) Expect to see most, if not all of…

- FORM
- MENU / SELECT
- TEXTBOX
- TEXTAREA
- BUTTON
    - SUBMIT, RESET
    - RADIO
    - CHECKBOX

b) No standard answer. Expect comments/phrases like …

- Simple, uncluttered
- Consistent
- Can see what to do next
- Don't strain user's short term memory
- Design for usability
- Can undo or get back from a wrong move

Examiner's Guidance Notes

Popular but frequently misunderstood.

a) Most answers were for 'what I know about websites', nothing about aspects with which a user can interact at all. Some clearly did not know what 'interact' actually means

b) Done better, but lots of unselective answers – they contained any aspect of website design and left the examiner to judge whether it represented good design or not.

Question B12

B12 Describe the general concept of white box testing and illustrate your answer using the function printed in question 4 or a similar sized function of your own choosing.

(12 marks)

Answer pointers

White box testing a process of testing a subroutine (procedure or function) with full knowledge of the code, so that the testing can guarantee to exercise all pathways through the code

So for the code of subroutine u, discover what part v, w play in code and choose values to test the code in 'normal' use and 'extreme' use.

check what happened when parameter v is set

- below subscript range for array
- in range for array (possibly at extreme bounds of array as well)
- above upper subscript range for array
- check what happens when parameter w is set
- below any value in array
- within the values for array
- above any value in array

Examiner's Guidance Notes

Often the last question attempted as many answers were clearly left unfinished. Many quoted function code without knowing how 'white box testing' was used to test it. Others wrote a comparison with Black Box testing.