BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

April 2011

EXAMINER'S REPORT

Software Development

General Comments

Many candidates were inadequately prepared for the programming questions. This means that a majority do not attempt these questions. The depth of ignorance revealed is sometimes quite profound.

Guidance

Avoid whole pages of crossing-out. It waste an inordinate amount of time.

Avoid writing answers in the wrong answer book. This is a nuisance as they have to be marked out of sequence with the majority of answers.

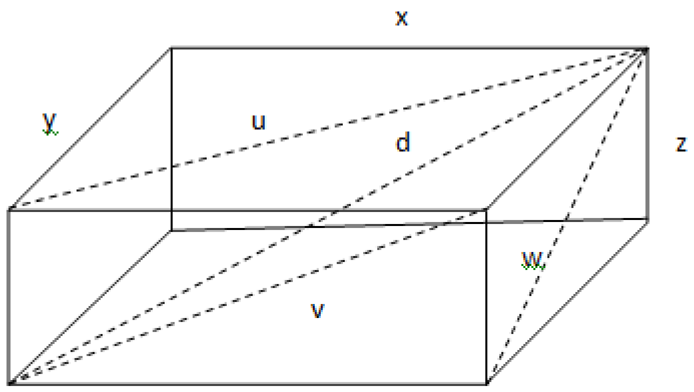
Do not copy the question into the answer book. This gains no marks and wastes time.

Declaration of records/pointer structures should not be copied into later parts of the answer but an appropriate place for them should be shown.

System commands associated with 'C' [uses crt, clrscr, #include <iostream.h>] are not needed – the examiner does not try out candidates' answers on a computer.

Section A

A1. A regular cuboid has sides (*x, y, z*), face diagonals (*u, v, w*) and body diagonal (*d*) as shown in the diagram below.



No cuboid is known to exist where all the values (*x, y, z, u, v, w, d*) are integers, but some combinations of integer values of *x, y, z* yield values of *u, v, w, d* which are close to integers.

a) Write a real function *side* which takes as parameters two integer values *base* and *height* and which returns the function value *side* calculated by Pythagoras' theorem.

$$side = \text{SQRT}(base^2 + height^2)$$

(2 marks)

b) Calculate the face diagonals *u, v, w* and body diagonal *d* using the function *side* and with the relevant base and height variables.

(4 marks)

c) Write a real function *diff* which takes one real value *diagonal_value* and returns the difference of this value from its nearest integer; that is,

diff = absolute value of (*diagonal_value – nearest integer to diagonal_value)*

Assume the values of *diff* are awarded points according to the table given below:

diff is:	Points awarded
less than or equal to 0.00001	50
greater than 0.00001 and less than or equal to 0.0001	20
greater than 0.0001 and less than or equal to 0.001	10
greater than 0.001 and less than or equal to 0.01	8
greater than 0.01 and less than or equal to 0.5	4

Write an integer function, *total_pts*, which has four real parameters representing the *diff* of *u, v, w*, and *d* and returns the total points awarded.

(8 marks)

d) When the maximum length of any side is *max_length*, and the minimum difference is given by *min_diff*, write the statements which

i) calculate all the diagonals using an appropriate loop structure
ii) print out the values *u, v, w, d* when the *diff* of all of these values is less than *min_diff*.

(6 marks)

e) For each *x, y, z* integer combination associated with a cuboid of *max_length*, choose an appropriate data structure to hold *x, y, z, u, v, w, d* and *total_pts*. Write a procedure *mysort()* with appropriate parameters to sort the data structure into descending order of points values.

(10 marks)

Answer Pointers

a)

```
REAL  FUNCTION side ( p, q as INTEGER)                                  (2 marks)
     side ←SQRT (p*p  +  q*q)
ENDFUNC
```

b)

```
u ← side(x, y)
v ← side(z, x)
w ← side(y, z)
d ← side(u, z)                                                          (4 marks)
```

c)

```
REAL  FUNCTION diff (val as REAL)
     diff = ABS(val – ROUND(val))
ENDFUNC

INTEGER FUNCTION pts (diff as REAL)
IF diff <= 0.00001 THEN pts ← 50
     ELSE IF diff <=  0.0001 THEN pts ← 20
          ELSE IF diff <=  0.001 THEN pts ← 10
               ELSE IF diff <=  0.01 THEN pts ← 8
                    ELSE  pts ← 4
                    ENDIF
               ENDIF
          ENDIF
     ENDIF
ENDFUNCT

INTEGER FUNCTION total_pts (u, v, w, d as REAL)
       total_pts = pts(diff(u)) + pts(diff(v)) + pts(diff(w)) + pts(diff(d))
ENDFUNCT                                                                (8 marks)
```

d)

```
FOR x ← 1 TO max_length DO
     FOR y ← x TO max_length DO
          FOR z ← y TO max_length DO
               IF (diff(u) < min_diff) AND (diff(v) < min_diff) AND (diff(w) < min_diff) AND (diff(d) <
               min_diff) THEN PRINT x, y, z, u, v, w, d, total_pts(u, v, w, d)
          ENDFOR
     ENDFOR
ENDFOR                                                                  (6 marks)
```

e) A suitable record structure is

```
     cuboid as RECORD
                    x, y, z, total_pts, as INTEGER
                    u, v, w, d as REAL
               ENDREC

     cubetype ARRAY [1..max_length*max_length*max_length]  OF  cuboid
```

Sort method. Potentially a lot of cuboid records may have to be sorted. Here a 'bubblesort' is used.

```
PROCEDURE mysort (RETURN VARIABLE X as cubetype, nitems as INTEGER)
(* bubblesort adapted for cuboid point ordering *)
VAR innerloop, outerloop, swapct as INTEGER
    swaps as BOOLEAN
    temp as CUBOID
BEGIN
    Swaps ←TRUE
    outerloop ← 0
    REPEAT
        swaps ← FALSE
        ADD 1 TO outerloop
        innerloop ← nitems
        WHILE innerloop > outerloop DO
            SUBTRACT 1 FROM innerloop
            IF X[innerloop].totpts < X[innerloop+1].totpts THEN
                (* SWAP ITEMS IN ADJACENT LOCATIONS *)
                Temp ← X[innerloop]
                X[innerloop] ← X[innerloop+1]
                X[innerloop+1] ← temp
                Swaps ← TRUE
            ENDIF
        ENDWHILE
    UNTIL swaps = FALSE
    PRINT "bubblesort ended."
ENDPROC
```

(10 marks)

Guidance Notes

Answers to this question were few and far between. Very many were poorly done and worth only single-figure marks. The examiner suspects that most candidates are not prepared for this standard of program development.

The question does NOT expect a complete program to be written from scratch. Candidates were only expected to write particular functions *side()* ,*diff()*, *total_pts*(), *mysort*() as described to be part of an overall design. Nor were variable declarations expected for auxiliary variables like *total*, *ct* and so on. Unfortunately after writing *side*, nearly all candidates departed from the question and went their own way towards writing a complete program. Naturally they didn't get there!

It is foolish to attempt an 'A' section question and know only part (a), say, worth 2 marks.

Nowhere does the question suggest that algebraic re-arrangements of the side terms is needed; some students wasted pages attempting to do this. The question stated that writing functions *diff()*, *total_pts*() and *mysort()* is expected. A complete program was not expected. It was very foolish to attempt to do this without any of the prescribed functions or any development.

Instructions like # Include <stdio.h> or # Include<conio.h> are not required in a question on algorithmic development.

A2. The following code is an iterative process for calculating the nth member of a Fibonacci series, as shown in the table below:

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

```
Line            code
  1    WRITELN( “input which member of series”)
  2    READ(n)
  3    p ← 2
  4    prev1 ← 1
  5    prev2 ← 1
  6    WHILE p IS LESS THAN n DO
  7    BEGIN
  8    term ← prev1  +  prev2
  9    prev2 ← prev1
 10    prev1 ← term
 11    p  ← p  +  1
 12    END
 13    WRITELN (“term  =”, term)
```

a) Dry run this code with n input as 4.

(18 marks)

b) Write code for a function to implement a <u>recursive</u> method for this process.

(6 marks)

c) Do you prefer the iterative style used in part a) or the recursive style used in part b)? Give reasons.

(6 marks)

Answer Pointers

Column headings for the table were awarded marks as shown below for variables that hold different values as the run progresses. Thus ‘n’ does not need a column as its value remains 4 throughout the run.

Line	Instructn	p	Prev1	Prev2	Term	Output	Other/remarks
…	…………….		……..	……..	…..		
	1 mark	1 mark		1 mark	1 mark	1 mark	1 mark

Subject to the proviso that not more than **5 marks** were awarded for headings.

a)

Line	instructn	p	Prev1	Prev2	term	Output	Other
1	WRITELN	?	?	?		Input..series	
2	assign						n= 4
3	assign	2					
4	assign		1				
5	assign			1			
6	WHILE						p<n:TRUE
7	BEGIN						
8	assign				2		
9	assign		1	1			
10	assign		2	1			
11	P = p + 1	3					
12	END						
6	WHILE						p<n:TRUE
7	BEGIN						
8	assign				3		
9	assign		2	2			
10	assign		3				
11	assign	4					
12	END						
6	WHILE						p<n:FALSE
13	WRITELN					term = 3	
14	END						Finish

b)
```
FUNCTION  recfib (n:INTEGER):INTEGER;

BEGIN
    IF n > 2 THEN recfib := recfib (n – 1)  +  recfib(n – 2)
                 ELSE recfib := 1
END;
```

c) The recursive method is simpler to code and closely resembles the definition. However, for a member well down the series (e.g. 1000) a long sequence of stack entries is generated which would use a lot of memory. So the iterative method may be preferred as the memory requirement is minimal.

Guidance Notes

This was a popular question; the full range of marks was used. Most candidates are familiar with the tabular form of answer, though other forms were not penalized.

a) It is desirable to have one column for ‘output’. Others got some way into the dry run table , then crossed it all out, sometimes as much as two pages of work. This would have wasted a lot of the candidate’s time. Candidates need to practice such answers without extensive rough work.

b) Very few candidates knew anything about recursion, they simply copied out the iterative code given. Just a few had different recursive code such as Fib((n-1) + (n-2)), not realising that this alters the parameters in the function declaration.

c) Any valid reasoning was given marks including subjective ideas as ‘recursive code is easier to understand’ and ‘iterative code uses less memory’.

A3. The array **x** has been initialised as follows

index	0	1	2	3	4	5	6	7	8	9	10	11
x	1	9	5	0	8	6	3	5	1	0	2	9

The function ***a*** in the code below is going to be executed with parameter ***b*** set to 1 and parameter ***c*** set to 10. [*You can choose to follow either version of the code*]

a) Trace the call of the function ***a(1,10)*** and show clearly the results of the call.

(8 marks)

	Version 1 (Pascal)	Version 2 (C)
1	float a(int b, int c){	FUNCTION a(b, c : INTEGER):REAL;
2	int d,e;	VAR d, e : INTEGER;
3	/* begin function */	BEGIN
4	d = 0;	d := 0;
5	e = b;	e := b;
6	while(e <= c)	WHILE e <= c DO
7	{	BEGIN
8	d += x[e];	d := d + x[e];
9	e++;	e := e + 1;
10	}	END;
11	return(d/(b-c+1))	a := d/(b-c+1)
12	}	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 1 or version 2] using your new names and including suitable comments.

(10 marks)

d) Rewrite lines 5 to 10 [of either version 1 or version 2] using a for-loop instead of a while-loop.

(6 marks)

Answer Pointers

(a) Trace of the call of the function ***a(1,10)***

```
line 1:    b=1, c=10
line 4:    d=0
line 5:    e=1
line 6:    1<=10
line 8,9: d+=x[1] => d=0+9=9; e++ => e=2
line 6:    2<=10
line 8,9: d+=x[2] => d=9+5=14; e++ => e=3
line 6:    3<=10
line 8,9: d+=x[3] => d=14+0=14; e++ => e=4
line 6:    4<=10
line 8,9: d+=x[4] => d=14+8=22; e++ => e=5
line 6:    5<=10
line 8,9: d+=x[5] => d=22+6=28; e++ => e=6
line 6:    6<=10
line 8,9: d+=x[6] => d=28+3=31; e++ => e=7
line 6:    7<=10
line 8,9: d+=x[7] => d=31+5=36; e++ => e=8
line 6:    8<=10
line 8,9: d+=x[8] => d=36+1=37; e++ => e=9
line 6:    9<=10
line 8,9: d+=x[9] => d=36+0=36; e++ => e=10
line 6:    10<=10
line 8,9: d+=x[10] => d=36+2=38; e++ => e=11
line 6:    11<=10 (false)
return ( 38/(10-1+1) ) =>  return 3.8
```

(b) the function finds the **average** of the numbers in the array x, **from index b, to c** inclusive

3*2=6marks

(c) better names
a => average, b => low, c => high, d => sum, e => index

```
float average(int low, int high){
   int sum, index;
   /* begin function */
   sum = 0;
   index = low;
   while(index <= high) /* step through array from index low to high */
   {
      sum += x[index]; /* add the current element to the total */
      index++;
   }
   return( sum/(high-low+1) ) /* divide sum by the count of elements to find average */
}
```

4 marks for variables
6 marks for comments

(d)

```
for(e = b; e <= c; e++){
        d += x[e];
}

for(index = low; index <= high; index++){
        sum += x[index];
}
```

Guidance Notes

The instruction on Line 11 “return(d/(c-b+1))” was printed as “return(d/(b-c+1))”. This was taken into account during the marking of the scripts.

a) Getting the 'right answer' is only worth half the marks. Many candidates didn’t show the appropriate traces. “d+=x[e]” or “d := d + x[e]” is the main line to be traced. Initial and the final values of e and d are important to show in the traces.

   A few candidates confused the subscript/index of an array with the contents at that subscript/index. For example, the value of x[1] is 9. It is NOT 8 or 0.

b) Precision was required. There are three points to be made. "The function *a* sums all the numbers in the array *x* between indexes ***b*** and ***c*** and divides the total sum by (c-b+1) in order to calculate the ***average***"

   The function does NOT sort the values in the array x.
   The function does NOT find the maximum value in the array x.
   The function does NOT find the minimum value in the array x.

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. a, b, c, d, e). Many candidates didn’t put any comments in their code. Marks were not given for superficial comments like 'start of function', 'start of loop', 'end of loop', 'end of function'.

d) This part was generally done well. A few candidates merely changed the word ' while' to 'for', hoping that would be enough - it wasn't. Many others put a reasonable condition after the *for*, but then forgot to remove the counting code (e=e +1;) from the main loop.

A4. Base your answers on either program version 1 or version 2 below.

Line No.	Version 1	Version 2
1:	/* program compareTest */	program compareTest;
2:	**int i** ;	**var i : integer** ;
3:	**char** c **;**	c : **char** **;**
4:	char **compare**(int p1,**p2**)	function **compare**(p1, **p2** : integer) : char ;
5:	{	begin
6:	if(p1 **==** p2)	if p1 **=** p2 then
7:	return(**'='**);	compare := **'='**
8:	else if(**p1 > p2**)	else if **p1 > p2** then
9:	return('>');	compare := '>'
10:	**else**	**else**
11:	return('<');	compare := '<'
12:	} **/* compare */**	end **/* compare */**
13:	void main(){	begin
14:	**i** **=** **2** ;	**i** **:=** **2** ;
15:	**c** = compare(**i-1** , 1 **+** 1);	**c** := compare(**i-1** , 1 **+** 1);
16:	**i = 3** ;	**i := 3** ;
17:	c = **compare(i, 1+1)** ;	c := **compare(i, 1+1)** ;
18:	}	end.

Key: description		
A: character constant	G: character identifier	M: variable declaration
B: integer constant	H: integer identifier	N: boolean (logical) expression
C: assignment operator	I: type identifier	O: formal parameter
D: arithmetic operator	J: function identifier	P: actual parameter
E: equality operator	K: reserved word	Q: function call
F: punctuation character	L: comment	R: assignment statement

a) Copy and complete the following answer table. (You need only copy out either the "Text in version 1" or the "Text in Version 2" column.) Then for each highlighted part in the programs above, decide what key it should have according to the table above.

(18*1 mark)

Line	Text in version 1	Text in version 2	Key letter for highlighted code
2	int i ;	var : i integer ;	
3	char	char	
3	;	;	
4	compare	compare	
4	p2	p2	
6	==	=	
7	'='	'='	
8	p1 > p2	p1 > p2	
10	else	else	
12	/* ... */	/* ... */	
14	i	i	
14	=	:=	
14	2	2	
15	c	c	
15	i-1	i-1	
15	+	+	
16	i = 3	i := 3	
17	compare(...)	compare(...)	

b) Find and copy out one example of each of the following:

i) arithmetic expression
ii) conditional statement
iii) function declaration

(3 x 4 marks)

Answer Pointers

(a)

Line	Text in version 1	Text in version 2	Key letters for highlighted code
2	v	v	Q
2	;	;	L
3	int	integer	F
3	order	order	I
3	c2	c2	N
4	int	integer	H
4	x	x	P
6	>	>	E
7	=	:=	C
7	1	1	A
8	c1 == c2	c1 = c2	M
9	x = 0	x := 0	R
11	-	-	D
10	else	else	J
13	/* ... */	/* ... */	K
15	'a'	'a'	B
16	v	v	G
16	w	w	O

(b)

i) arithmetic expression
```
        i-1 or  1 + 1
```

ii) conditional statement
```
   if( p1 > p2 )
      return('>');
   else
      return('<');
```

iii) function declaration

```
char compare(int p1,p2)
{
   if( p1 == p2 )
      return( '=' );
   else if( p1 > p2 )
      return('>');
   else
      return('<');
}
```

Guidance Notes

This was the most popular question of section A, but often poorly answered.

(a)
The instructions to candidates were clear - they were to copy out and complete a table for part (a). Those who did not do this were disadvantaged because their own choice of answer format was often not precise enough to gain marks.

Candidates should take more care to write clearly. In many cases in this question getting the marks depends on the examiner being able to read a single letter confidently.

It was surprising to see candidates choosing to answer this question and then getting all 18 parts of part (a) wrong.

It seems that candidates cannot distinguish between:
- a character constant *versus* a character identifier
- an integer constant *versus* an integer identifier
- an assignment statement *versus* an assignment operator
- a formal parameter *versus* an actual parameter
- an arithmetic operator *versus* an equality operator

Some candidates could not even identify a 'comment'

(b)
The instruction was to copy from the question, not invent parts of a program. The instruction was to copy from the question not give definitions of an "arithmetic expression", etc.

(i) It was important not to write too much. For example, "i-1" is an arithmetic expression and a CORRECT answer but "c:=compare(i-1,1+1);" is not an arithmetic expression (although it contains arithmetic expressions) and is thus a WRONG answer

(ii) All the "if" statements in the question have "else" parts. Therefore if you are copying out an "if" statement from the question, you must copy it all, including the "else" part.

(iii) Many answers did not copy out the whole function declaration, just the function heading. This was awarded half marks.

SECTION B

B5. a) Write a definition for a RECORD (or STRUCTURE) to hold information about a property for sale. The details to be held are: price (up to 6 digits whole number), type (house, flat or bungalow), garage (yes/no), number of bedrooms (small integer), address (3 lines, each of 20 characters).

(4 marks)

b) Write a FILE declaration for a file called 'propfile' to hold a sequence of these records.

Write code which requests an upper price limit, and which then searches the file for all properties below this limit with a garage and 3 or more bedrooms. The address of each property satisfying these conditions is to be printed. If no property is found, the message 'no such property available' should be printed.

(8 marks)

Answer Pointers

a)
```
Property  =  RECORD

                  Price  :  LONGINT
                  Type  :  (house, flat, bungalow)
                  Garage  :  BOOLEAN
                  Bedroom  :  SMALLINT
                  Address  :  ARRAY[1..3] OF ARRAY[1..20] {or string}
              ENDREC
```
(4 marks)

b)
```
Propfile  :  FILE OF  property

Program searchprop (INPUT,OUTUT);
VAR
BEGIN
    WRITELN('property search program');
    WRITELN('input UPPER property price limit');
    READ(limit);
    OPEN propfile for READING
    count := 0;
    WHILE NOT EOF(propfile) DO
    BEGIN
        READ(propfile,onerec)
        WITH onerec DO
            IF price < limit THEN
                IF garage  AND  bedrooms >=3  THEN
                BEGIN
                    FOR n := 1 TO 3 DO WRITELN(address[n])
                    count := count + 1
                END
        ENDWITH
    END
    IF count = 0 THEN WRITELN('no such property available')
    CLOSE(propfile)
END.
```

(8 marks)

Guidance Notes

A popular question, most answers had only the record declaration. Very few knew about condition controlled loops needed here. A worse mistake was including the ellipsis (…) in code. Answers usually contained [uses crt, clrscr, #include <iostream.h>]. Many gave test data for streets which was not asked for.

Some students gave a linked list diagram. While some systems implement files in this way, they are usually different in how they are used and coded

B6. a) i) Many programming languages contain a statement which allows one action to be selected from a group of actions based on a value held in a variable. Illustrate this with a statement from a language of your choice.

(2 marks)

ii) Give its equivalent in terms of IF…THEN…ELSE statements.

(2 marks)

b) A Government wishes to implement a Wealth Tax according to the Assets in the following table:

Assets (pounds)	Assets integer divided by 100,000	Tax (percentage)
Less than 100,000	0	0
100,000 to 399,999	1, 2, 3	0.05
400,000 to 699,999	4, 5, 6	0.1
700,000 to 999,999	7, 8, 9	0.15
Greater than 1,000,000	10	0.2

i) Write an expression relating Assets (pounds) to Tax (percentage).

(3 marks)

ii) Incorporate your expression in a program or pseudocode which receives as input a particular value for assets (for example 587,000 pounds) and which outputs the corresponding value of the tax payable in pounds.

You should not write out again your expression for i) but indicate where it belongs in your program.

(5 marks)

Answer Pointers

a) Expect a Case statement or similar (2 marks)

b) Expecting nested IF statements (2 marks)

c) i) A relationship exists between the asset values and a sequence of integer values: Dividing each by 100,000 gives the integers in the middle column. This can be seen quite easily as the groups of values each contain 3 integers.

```
CASE Assets DIV 100000 OF
        0 : wtax  ← 0.0
        1, 2, 3 : wtax ← 0.05
        4, 5, 6 : wtax ← 0.10
        7, 8 9 : wtax  ← 0.15
      OTHERWISE
         wtax  ← 0.2
ENDCASE
```

(3 marks)

ii)

```
PRINT " input the value of the asset in pounds"
INPUT Asset
Case statement code here
Pay_tax := Asset * wtax / 100
PRINT "tax payable = "  Pay_tax:6:2
END.
```

(5 marks)

Guidance Notes

Very few knew what the statement was. Most students wrote out the 'case' code twice despite being told not to.

B7. Conversion of a decimal number to its binary equivalent is carried out as follows:

The decimal number is successively divided by 2 and the remainders are stored. The remainders in reverse order form the binary number; the last remainder being the most significant digit. Thus 13 (decimal) = 1101 (binary)

Process	Quotient	remainder	
÷2	13	1	(least significant bit)
÷2	6	0	
÷2	3	1	
÷2	1	1	(most significant bit)
	0		

Write pseudocode or a program to input a decimal number and output its binary equivalent. Only integer decimal numbers are to be considered.

(12 marks)

Answer Pointers

```
PROGRAM decbin(INPUT,OUTPUT);
    {Prints table of decimal - Binary equivalents}
    TYPE binno = 0..1;
    VAR pos, inner, decno, rem : INTEGER;
          unfinished:BOOLEAN;
          bit:ARRAY[1..16] OF binno;

    BEGIN {top level}
        PRINT “DECIMAL TO BINARY CONVERSION PROGRAM');
        {decimal to binary by repeated division & store remainder}
        WRITELN(“input decimal number”)
        READ ( decno )
        pos := 1
        unfinished := TRUE
        WHILE unfinished DO
        BEGIN
          rem := decno MOD 2
          bit[pos] := rem
          pos := pos + 1
          decno := decno DIV 2
          unfinished := decno > 0
        END
        FOR inner := pos - 1 DOWNTO 1 DO WRITE(bit[inner]:1)
    END
END.
```

Guidance Notes

Few had any idea that a decimal input was expected, followed by a condition-controlled loop; they merely copied out the example concerned with decimal 13.

B8. a) i) Show diagrammatically a linked list of 4 nodes, where each node consists of one pointer and one integer data item.

(2 marks)

ii) Give a declaration of this data structure in a programming language of your choice. State the language you are using.

(2 marks)

iii) State, with a reason, an application for which this particular data structure is useful.

(2 marks)

b) i) Show diagrammatically a one-dimensional array with one subscript.

(2 marks)

ii) Give a declaration of this data structure using the same programming language as in part a).

(2 marks)

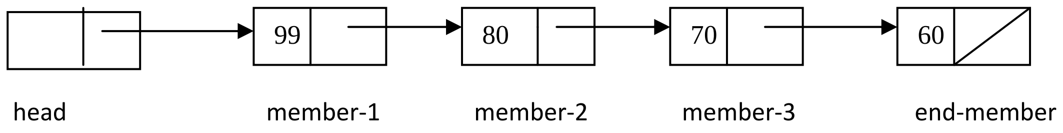
iii) State, with a reason, an application for which this particular data structure is useful.

(2 marks)

Answer Pointers

a)

i) Linked list



(2 marks)

```
ii) TYPE     ptr = ^node;
             node =   RECORD
                         data : INTEGER;
                         next : ptr
                      END;
```
(2 marks)

iii) This structure is useful for storing a sequence of related data items where the <u>number of these is not known initially</u> and which <u>changes dynamically</u> as the run progresses. **(2 marks)**

b)

i)

99	80	70	60

contents:

```
                     ↑         ↑         ↑          ↑
subscript           [1]       [2]       [3]        [4]
```

(2 marks)

ii) smallarray : ARRAY[1..4] OF INTEGER; **(2 marks)**

iii) This data structure is particularly useful in SORTING operations when all of these are in memory at the outset. The data in two locations can be swapped using the subscripts thus:

```
IF item[subscr-1} IS GREATER THAN  item[subscr-2]
            THEN swap (item[subscr-1], item[subscr-2])
```

The values of the two subscripts [subscr-1, subscr-2] can be varied systematically and data between any pair of locations in the array can be swapped. **(2 marks)**

Guidance Notes

This question was usually well done. Most students were weak on areas of application for the data structures.

B9. One of the standard data structures used in programming is the *stack* with its distinctive operations of *push* (place a new value on the *stack*) and *pop* (remove the top value from the *stack*). Using an array as the basic storage for the *stack*, write code in a language of your choice to implement the operations *push* and *pop*.

(12 marks)

Answer Pointers

This answer is in the programming language C. Other languages were accepted

```
int stack[100]; /* array to store stack */
int top; /* stack top pointer */
top=0;

void push(int v){
        stack[top++]==v;
}

int pop(){
        return( stack[--top] );
}
```

setup 3 marks, push 3 marks, pop 3 marks, attention to stack overflow/underflow 3 marks

Guidance Notes

The code for stack overflow/underflow is not shown above. It is simple to add but note that many answers went wrong by writing

```
      if( stack[top] > ?? )...overflow
or    if( stack[top] < ?? )...underflow
```

when it should have been

```
      if( top > ?? )...
or    if( top < ?? )...
```

Why were the stacks in the solutions so small? e.g. array[1..3]
Why were there loops in the stack functions?
Why (despite the very clear instruction in the question) were the stack built on lists rather than arrays?

B10. Give one reason why someone might prefer:

a) a compiler *rather than* an interpreter
b) a command line operating system *rather than* a WIMP operating system
c) a 4GL programming language *rather than* a 3GL language
d) a phased implementation of a new system *rather than* a parallel implementation
e) a low-level language *rather than* a high-level language
f) an array *rather than* a linked list

(6 x 2 marks)

[Note: It is expected that one well-chosen sentence per part will be sufficient as an answer]

Answer Pointers

a) compiler: because the compiled code runs faster
b) command line: because of greater control, scripting possibilities
c) 4GL: because of speed/convenience
d) phased: because of the overhead of doing everything twice over in parallel
e) low-level: because of device/memory restriction e.g. tiny memory in embedded chip
f) array: because it is simpler

Guidance Notes

A very popular question. Often the quality of the 6 parts was very uneven. Many candidates didn't give a reason why someone might prefer the terms given in the questions. They only wrote the meaning of the terms.

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 processes 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) A few candidates explained why a WIMP operating system is better than a command line OS. They should have explained why a command line operating system is better than WIMP OS. This part was done quite badly.

c) This part was done well. However A few candidates gave wrong examples. For example, assembly language is a NOT 3GL.

d) This part was done badly. Many candidates explained software development models, such as Waterfall model. A few candidates described the concepts of the parallel processing.

e) This part was done well. A few candidates didn't understand the question and explained why they prefer a high-level language rather than a low-level language. Some candidates gave wrong examples. Java is NOT a low-level language.

f) This part was done very well. Many candidates explained the advantage of an array over a linked list.

B11. While working on a prototype of a new system you require a web page for a new user to register. The page should contain:

i) a way for the new user to enter their name
ii) a way to enter the range containing the age of the user (0-9, 10-19, 20-29, etc)
iii) a way to enter the gender of the user
iv) a way for the user to show which by combination of methods (phone, email, letter) they are willing to be contacted
v) a way to announce that the data is ready to be received by the system

a) Draw a diagram to illustrate your idea for the interface. The page should contain a form with appropriate elements to handle the various types on information.

(5 x 2 marks)

b) Pick one of the pieces of information that is to be entered, state an alternative method that could have been used to enter that information and state why it would be inferior to the method used in a).

(2 marks)

Answer Pointers

a) Diagram should show
i) a way for the new user to enter their name (TEXT BOX)
ii) a way to enter the range containing the age of the user (0-9, 10-19, 20-29, etc) (MENU)
iii) a way to enter the gender of the user (RADIO BUTTONS)
iv) a way for the user to show which by combination of methods (phone, email, letter) they are willing to be contacted (CHECKBOXES)
v) a way to announce that the data is ready to be received by the system (SUBMIT BUTTON)

b) [Just need ONE of these]

Alternative to (i) might be text area - but don't need multiline for name
Alternative to (ii) might be radio button - but it would take up a lot of space
Alternative to (iii) might be menu - but that is usually reserved for cases when there are more alternatives
Alternative to (iv) might be a multi-choice menu - but these are not common/popular
There is no real alternative to (v)

Guidance Notes

Note that the question centres around the parts of the user interface where the user gives input to the system. This means that when the 5 interaction elements required by part (a) had been drawn, there were no more marks for candidates who decided that the page should have more on it and often drew far too much.

Part (b) was often misunderstood. It was not an invitation to criticise the design of the whole page (several candidates discussed various ways to enter postal addresses - or email addresses - or phone numbers - or the actual age of the user - none of these was included in the question). It was simply to choose from the elements already in part (a) and give an alternative and to say why the alternative would not be such a good solution.

B12. Choose program version 1 or 2 below and then find occurrences of the following 6 errors. For each error, you should give the line number and an explanation of the error. In addition, you should state whether each error will be discovered at compile time or run-time.

(6 x 2 marks)

i) identifier not declared
ii) type error - index not allowed
iii) array index out of bounds
iv) syntax error
v) variable required
vi) type error - invalid type

line	Version 1 (Pascal)	Version 2 (C)
1	program p;	void main();
2	var a : array[0..9]of real;	float a[10];
3	b, c : integer;	int b, c;
4	begin	{
5	b := 1;	b = 1;
6	2 := b;	2 = b;
7	c := b[1];	c = b[1];
8	d := b+c;	d = i+j;
9	if b > 1 b := b - 1 ;	if (b > 1 b = b - 1 ;
10	a[0] := b;	a[0]=b;
11	if a[0] then c := 1;	if (a[0]) c = 1;
12	for b:=0 step 1 to 9 do	for(b=0; b<=9; b++)
13	a[b+1]:=b;	a[b+1] = b;
14	end.	}

Answer Pointers

i) line 8: "d" not declared [compile time]
ii) line 7: index not allowed, in b[1] b is not array [compile time]
iii) line 13: last time round loop, b is 9, b+1 is 10, so a[b+1] is invalid [run time]
iv) line 9: "then" missing from Pascal, ")" from C, [compile time]
v) line 6: variable required (in place of 2) [compile time]
vi) line 11: boolean expression required after if (not float/real) [compile time]

Guidance Notes

This was a popular question, but not necessarily answered very well. Many candidates didn't state whether each error will be discovered at compile time or run-time.

i) Many candidates stated that variable *d* needs to be declared before it is used. However the line number of the error "identifier not declared" is not line 2 or 3.

ii) Candidates generally stated that *b* is not an array and there will be an error on line 7. However, some candidates wrongly stated that it is an invalid type error.

iii) The error - "Array index out of bounds" was generally spotted and explained well. The line number of the error is wrongly stated 12. A few candidates wrongly stated that "Array index out of bounds" is a compile-time error.

iv) This error is spotted and explained very well.

v) Line 6 error is spotted well but usually explained wrongly. The error is not a syntax error. The assignment is expecting a variable.

vi) This error is rarely spotted. Candidates wrongly state that Line 11 has a syntax error.