BCS HIGHER EDUCATION QUALIFICATIONS
BCS Level 4 Certificate in IT

September 2011

EXAMINER'S REPORT

Software Development

Important Announcement:

The inclusion of Pascal extracts in the examination paper is being phased out and will not appear after April 2012. In future papers, where a programming language is used, questions will be set using examples in C (only) which should be easily understandable by users of Java, C++, PHP, Javascript, etc

Answers to programming questions will continue to be accepted in any reasonable language (including Pascal), but C and languages based on the syntax of C are recommended.

General:

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.

Regular marked homework is a better preparation for the programming questions than practical work.

System commands associated with 'C' [uses crt, clrscr, #include <iostream.h>] are not needed.

Section A

A1. The triangular table shown below shows a sequence of railway stations (such as A, B, C, D, E, F) on one line and the distances in kilometres (km) between all of them. For example,

	A	B	C	D	E	F
A	0					
B	5	0				
C	15	10	0			
D	20	15	5	0		
E	30	25	15	10	0	
F	35	30	20	15	5	0

In this example, the distance between stations E and C is 15km and between C and F is 20km.

a) Assuming a station name can be no longer than 40 characters, that each station has a unique 3 character code identifier and there are a maximum of 200 stations, define a suitable data structure to hold this information. The unique 3 character code identifier is not the same as any station name.

(3 marks)

b) Assume an integer array holds the distances between the first station and each of the other stations. In this example, it would be the distance between A & A, A & B, A & C, A & D, etc. and the first four values would be 0, 5, 15, and 20. Write a sub-program which

i) takes as its parameters the data structure you defined in a), an integer array holding the distances from the first station and each other station and an integer which gives the number of stations and
ii) prints out the triangular table of distances using the three character code identifier. (NOTE – You can assume the table will fit on the output medium. You do not have to worry about lines "wrapping around".)

(12 marks)

c) Write an integer function which

i) has four parameters; namely two parameters which can be either two station names, two unique 3 character codes or one station name and one unique 3 character code, one parameter representing the data structure you defined in part a) and an integer parameter which gives the number of stations and
ii) returns the distance between the two stations passed to the function.

If either of the station names or unique character codes does not exist, then the function returns a negative value.

(15 marks)

Answer Pointers

a) For this example, a suitable data structure could be an array of records. (3 marks)

b)
```
procedure print_table( stations ARRAYRECS, distances ARRAYINTS, nos_stations INT)
begin
# declarations
CONSTANT      apart = 6;
VAR       i, j : integer;    # loop variables

# print out station codes every 6 characters say
print(" ":apart);
for i ← 1 to nos_stations do
begin
    print(stations[i].station_code:apart)
end;
printnewline;
# print out distances using two loops as shown below
for i ← 1  to  nos_stations do
begin
    print(stations[i].station_code:apart)
    for j ← 1 to i  do
    begin
# candidates have to work this out.
        print((stations[i].distance- stations[j].distance):apart)
    end
    printnewline
end
```

(12 marks)

c) **integer function** find_distance (station_a, station_b **CHAR_SEQN,** stations **ARRAYRECS**,
nos_stations **INT)**

```
begin
# declarations
    integer not_found = -1;
    integer dist_a, dist _b;
    integer function locate_station (station CHAR_SEQN, stations ARRAYRECS, nos_stations INT)
    begin
        boolean found;
        int posn;
        found ← false;
        posn← 1 :
        while not found do
        begin
# locate station
            if (stations[posn].station_name = station) or (stations[posn].station_code = station) then
                begin
                    found := true;
                end
            else
                begin
# if not found, check if any more stations
                    if posn < nos_stations then
                        begin
                          posn ← posn + 1     # look at next station
                        end
                    else
                        begin
                          found ← true;                 # no more stations; set to true to exit while loop
                          posn ← not_found
                        end;
                end;
        end;
        locate_station ← posn
    end; # of locate_station function

    dist_a ← locate_station(station_a, stations, nos_stations);
    dist_b ← locate_station(station_b, stations, nos_stations);

    if (dist_a = not_found) or (dist_b = not_found) then
        begin
            find_distance ← not_found
        end
    else
        begin
            find_distance ← abs(stations[dist_a].distance-stations[dist_b].distance)
        end;
end;   # end of find_distance function
```

Examiner's Guidance Notes

A poorly attempted question. Very few candidates showed an ability to break the problem down into smaller parts.

A2. A text file <intext> contains words separated by a space. The words contain only capital letters ['A'..'Z']. A sentence is terminated by a full stop.

The algorithm below is a first attempt to process the text to form an output file <outfile>. Letters are replaced by their lower-case counterparts, except the first one in any sentence. Other characters are copied to <outfile> unchanged.

```
Line                                    code
 1    OPEN <intext> for reading, <outtext> for writing
 2    first_char ← TRUE
 3    WHILE NOT EOF(intext)  DO
 4        READ(intext, achar)
 5        IF achar contains a capital letter AND NOT first_char THEN
 6            BEGIN
 7                convert contents of achar to a lower-case letter
 8                first_char ← FALSE
 9            END;
10        IF achar = stop  THEN first_char ← TRUE
11        WRITE(outtext, achar)
12    ENDWHILE
13    WRITELN("finished")
```

a) Write out the code for the test of "achar contains a capital letter" on line 5 and "convert contents of achar to lower-case letter" on line 7.

(5 marks)

b) Dry run the code with the five character sequence *G. TH* read from <intext>. Note – In the sequence there is a space character between the full stop and the T.

(20 marks)

c) What conclusions do you draw from the dry run? Suggest ONE amendment.

(5 marks)

Answer Pointers

a) Capital letters lie between a start-value = ORD('A') and [start-value + 26] = ORD('Z'), so a range check can be performed thus:

IF (ORD(achar) <= ORD('Z')) AND (ORD(achar) >= ORD('A')) THEN achar contains a letter

```
Num ← ORD(achar)
Change ← Num + (ORD('a') – ORD('A'))
achar  ←  CHR(Change)        `
```

(5 marks)

b) Dry Run

Line	Instructn	first_ char	Achar	infile	outfile	Output	Other
1	OPEN files	?	?	readi ng	writing		
2	Assign	True					
3	WHILE						NOT EOF (infile) → True
4	READ		G	G			
5	IF						True AND False→False
10	IF(.)						False

11	WRITE				G		
12	ENDWHILE						False
3	WHILE						NOT EOF(infile) →True
4	READ		Stop	Stop			
5	IF						False AND False→False
10	IF(.)	True					
11	WRITE					Stop(.)	
12	ENDWHILE						False
3	WHILE						NOT EOF(infile) →True
4	READ		Space				
5	IF						False AND False →False
10	IF(.)	Fals e					False
11	WRITE				Space		
12	ENDWHILE						False
3	WHILE						NOT EOF(infile) → True
4	READ		T	T			
5	IF						True AND False→False
10	IF(.)						False
11	WRITE				T		
12	ENDWHILE						False
3	WHILE						NOT EOF(infile) → True
4	READ		H				
5	IF						True AND False → False
10	IF(.)						False
11	WRITE		H		H		H NOT converted
12	ENDWHILE						EOF(infile) → True
13	WRITELN					Finished	

(20 marks)

c) It shows that the conversion H to h does not take place, as required in the specification. This is because the variable first_char is not reset to FALSE after the first chatacter (G) is immediately followed by a full stop (.) Different test data such as AB.T would be needed to show this to explore where the conversion B → b does actually happen.

A correction to Line 10 should be used to test if there was one or more spaces after the full stop.

10 IF (achar = ".") OR (achar = space) THEN first_char ← TRUE

(5 marks)

Examiner's Guidance Notes

a) Most candidates copied out what was given in the question without any development.

b) Most students tackled this quite well although few managed a completely correct dry run.

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 10 and parameter ***c*** set to 6. [*You can choose to follow either version of the code*]

a) Trace the call of the function ***a(10,6)*** and show clearly the results of the call.

(8 marks)

	Version 1 (C)	Version 2 (Pascal)
1	int a(int b, int c){	FUNCTION a(b, c : INTEGER):INTEGER;
2	int d,e;	VAR d, e : INTEGER;
3	/* begin function */	BEGIN
4	e = -1;	e := -1;
5	d = 0;	d := 0;
6	while(d < b)	WHILE d < b DO
7	{	BEGIN
8	if(x[d] == c)	IF x[d] = c
9	e = d;	THEN e := d
10	d++;	d := d + 1;
11	}	END;
12	return(e)	a := e
13	}	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 11 [of either version 1 or version 2] using a for-loop instead of a while-loop.

(6 marks)

Answer Pointers
a) Trace

a(10,6) means that b=10, c=6

e=-1
d=0

step	b	c	d	e	d<b	x[d]==c
	10	6	0	-1	y	n
	10	6	1	-1	y	n
	10	6	2	-1	y	n
	10	6	3	-1	y	n
	10	6	4	-1	y	n
	10	6	5	-1	y	y
				5		
	10	6	6		y	n
	10	6	7	5	y	n
	10	6	8	5	y	n
	10	6	9	5	y	n
	10	6	10	5		

return(5)

Notes: Either the result or the trace of the function call is mostly wrong or missing in the student answers. Trace of the variable “e” was usually confused with the value of the variable “d” since the statement “if(x[d] == c)” wasn’t considered appropriately. For a few candidates, it was difficult to read the value from the array.

b) Function 'a' visits 'b' elements of array x starting from index 0 and records the subscript at which it finds the value 'c'. The subscript is returned as the result. A value of -1 is returned if no value matching 'c' is found.

Notes: Candidates mostly gave good answer for this part. Some candidates explained how the subroutine works rather than writing the brief summary about it. Some wrote the explanation which is irrelevant to the code. For example this code doesn’t find the average of the numbers in the array.

c) Marks for 5 new names
and re-write with at least 2 meaningful comments

```
int search(int length, int searchItem){
/* function  to search part of array x for an item*/
    int i, positionFound;
    positionFound = -1; /* default if not found */
    i = 0;
    while(i < length)
    {
        if( x[i] == searchItem )
            positionFound = i;
        i++;
    }
    return( positionFound  ) /* array index */
}
```

Notes: Candidates mostly replaced variable names with new ones. However they used variable names which were no better than given one (e.g. “e” replaced with “x”). Some candidates have used meaningful

names but wrong names. For example the subroutine name “a” was replaced with “average” instead of “search”. Many candidates didn’t add comments into the code.

d) Need to recognise the initialisation of the counter; the continue/stop test; and the increment

```
for(d = 0; d < b; d++){
    if( x[d] == c )
        e = d;
}
```

Notes: Many candidates answered this part well. Common mistakes were the places of the initialization and incrementing of the “d” value (i.e. index). The “d++” should not be in between “{ }”. The “d=0” should not be before the for-loop.

Examiner’s Guidance Notes

See Note/Notes in the Answer pointers.

Overall, this question is very popular and poorly answered. Some students did not understand the code given at all. In part “a” their traces of the running program did not match the code. In part “b”, the students wrote how the program works or explained the benefits of functions, instead of a brief summary of what the function does. In part “c”, variable names used were not usually suitable with the purpose of the code.

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

Line No.	Version 1	Version 2
1:	/* program compareTest */	program orderTest ;
2:	int v , w ;	var v , w : integer ;
3:	int order(char c1, c2)	function order(c1, c2 : char) : integer ;
4:	int x ;	var x : integer ;
5:	{	begin
6:	if(c1 > c2)	if c1 > c2 then
7:	x = 1 ;	x := 1
8:	else if(c1==c2)	else if c1=c2 then
9:	x = 0 ;	x := 0
10:	else	else
11:	x = - 1 ;	x := - 1 ;
12:	return(x) ;	order := x
13:	} /* order */	end /* order */
14:	void main( ) {	begin
15:	w = 'a' ;	w := 'a' ;
16:	v = order(w , 'A') ;	v := order(w , 'A') ;
17:	}	end .

Key: description		
A: integer constant B: character constant C: assignment operator D: arithmetic operator E: relational operator F: result type	G: integer identifier H: type identifier I: function identifier J: reserved word K: comment L: punctuation character	M: boolean (logical) expression N: formal parameter O: actual parameter P: local variable Q: global variable 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 x 1 marks)

Line	Text in version 1	Text in version 2	Key letters for highlighted code
2	v	v	
2	;	;	
3	int	integer	
3	order	order	
3	c2	c2	
4	int	integer	
4	x	x	
6	>	>	
7	=	:=	
7	1	1	
8	c1 == c2	c1 = c2	
9	x = 0	x := 0	
11	-	-	
10	else	else	
13	/* ... */	/* ... */	
15	'a'	'a'	
16	v	v	
16	w	w	

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

i) function call
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 - global variable
2	;	;	L - punctuation character
3	int	integer	F - result type
3	order	order	I - function identifier
3	c2	c2	N - formal parameter
4	int	integer	H - type identifier
4	x	x	P - local variable
6	>	>	E - relational operator
7	=	:=	C - assignment operator
7	1	1	A - integer constant
8	c1 == c2	c1 = c2	M - boolean expression
9	x = 0	x := 0	R - assignment statement
10	else	else	J - reserved word
11	-	-	D - arithmetic operator
13	/* ... */	/* ... */	K - comment
15	'a'	'a'	B - character constant
16	v	v	G - integer identifier
16	w	w	O - actual parameter

Notes: Quite a lot of good answers, but equally many answers showing no understanding of the terminology used. For example, there was confusion between an assignment operator and an assignment statement. They are not the same thing. Candidates should be able to understand that an assignment operator appears *in* an assignment statement and that a relational operator can appear *in* a boolean expression. There even seems to be confusion between constants and variables.

b)
i) function call

```
order(w,'A');
```

Note: it was a mistake to add the "v=" in front of the call as was offered in many answers

ii) conditional statement

```
if( c1==c2 )
        x = 0 ;
else
        x = - 1 ;
```

Note: a) it is a mistake to copy too much e.g. "adding return(x);" at the end
b) it is a mistake to copy too little e.g. ending at the word "else"
c) return(...)" is not a function call

iii) function declaration

```
int  order( char c1, c2 )
int x ;
{
   if( c1 > c2 )
       x = 1 ;
   else if( c1==c2 )
      x = 0 ;
   else
      x = - 1 ;
   return( x ) ;
}
```

Note: main(...){...} is a function declaration in C and was accepted as a correct answer, but the main program is not a function in Pascal.

Note: Some answers made a bad mistake by reversing (i) and (iii)

Examiner's Guidance Notes

See Note/Notes in the Answer pointers

Some students did not read - or did not understand - the question and wrote entirely wrong answers - some trying to define a function (what it is and what it is for) - some giving examples but making them up, not copying from the question as required.

SECTION B

B5. a) Most programming languages provide a means of selecting one action from a group based on the value held in a variable. However, this value cannot be of type REAL. Explain why this is so.

(3 marks)

b) Develop an algorithm which reads a four-digit integer *Date* in the form mmdd (mm = month digits, dd = day digits) and assigns the appropriate season to a variable Time_of_Year. For example, if the *Date* was 1123 (that is, 23rd November) then "AUTUMN" would be assigned to Time_of_Year.

(9 marks)

Season	**Dates**		
SPRING	21 March	to 20 June	inclusive
SUMMER	21 June	to 20 September	inclusive
AUTUMN	21 September	to 20 December	inclusive
WINTER	21 December	to 20 March	inclusive

Answer Pointers

a) The resulting value on which the choice is based must be capable of being held exactly in pure binary. Real numbers are not like this, however they are implemented. **(3 marks)**

b)

Season	**Dates**		
SPRING	21 March	to 20 June	inclusive
SUMMER	21 June	to 20 September	inclusive
AUTUMN	21 September	to 20 December	inclusive
WINTER	21 December	to 20 March	inclusive

Declare variables
INPUT Date
mm ← Date DIV 100 dd ← Date MOD 100
assign season using dd and mm
PRINT season

Final Algorithm
STYPE = (Spring, Summer, Autumn, Winter)
PRINT "input date as <ddmm> ie 1412 as 14th of December"
INPUT Date
mm ← Date DIV 100
dd ← Date MOD 100
IF (dd <21) AND ((mm MOD 3) = 0) THEN SUBTRACT 1 FROM mm
CASE mm MOD 3 OF
    4,0 : Time_of_Year ← Winter
    1 : Time_of_Year ← Spring
    2 : Time_of_Year ← Summer
    3 : Time_of_Year ← Autumn
ENDCASE **(9 marks)**

Examiner's Guidance Notes

a) Very few candidates understood what was required here, usually the 'case' statement or equivalents from other languages. Almost no candidates realized why the selector cannot be of type real.

b) Most answered this with a nest of 'IF' statements with varying accuracy.

B6. A number is perfect if the sum of its factors including 1 equals the number. Thus 6 is perfect as the factors of 6 (that is, 3, 2 and 1) when added together equal 6.

Write a program or pseudocode which prints out all the perfect numbers between 1 and an input limit *limit.*

(12 marks)

Answer Pointers

```
PROGRAM perfect (INPUT,OUTPUT);
{program finds perfect numbers between 1 and an input upper limit}
VAR upper, sum, outer, inner, limit : INTEGER;
WRITELN('input upper limit for calculations');  READ(upper);
FOR outer := 2 TO upper DO
BEGIN
    sum ← 1
    limit ← outer  DIV  2
    FOR inner ← 2 TO limit DO
    BEGIN
        IF outer MOD inner = 0 THEN    {factor found}
                sum ← sum + inner{add to sum of factors}
    END;
    IF sum = outer THEN WRITELN('perfect number', outer)
END.
```

(12 marks)

Examiner's Guidance Notes

Few candidates understood how to find the factors of a number using 'IF number-1 MOD number-2 = 0 THEN number-2 is a factor of number-1'. A loop is also needed to run between 2 and limit

B7. The REAL function RANDOM() returns a value 0.0 < RANDOM() <=1.0.

a) Write an expression which uses RANDOM() and gives an integer value between 1 and 6, as appropriate for a dice score.

(3 marks)

b) Incorporate this expression in pseudocode or a program which accepts an integer input to the variable *howmany*, and then carries out *howmany* simulations of a dice throw. These are stored in a suitable data structure. The mean value of all the throws is worked out and compared to the known value of 3.50. Separately add up all the scores for dice throw 1,2,…6.

(9 marks)

Answer Pointers

a) score ← (RANDOM() * 5 + 0.5) + 1 (*integer random number between 1 and 6 1<=score<=6*)

b)

Developed algorithm

```
tot ← 0
FOR K ← 1 TO 6 dice[K] = 0                  (* set score totals to zero*)
INPUT how many simulations → howmany
FOR K ← 1 TO howmany DO                          (* main loop for simulations *)
        score ← ROUND(RANDOM() *6 + 0.5) MOD 6    (*integer random number between 1 and 6 *)
        dice[score] ← dice[score] + 1          (*increment appropriate score count*)
        ADD score TO tot
ENDFOR
PRINT  counts of score values caption
FOR K ← 1 TO 6
        prob ← dice[K] / howmany
```

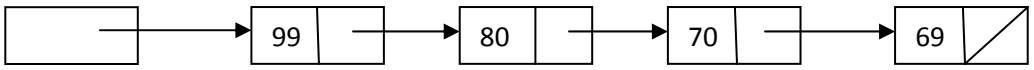
```
        PRINT  K, dice[K],  prob                (* score, how many times it was produced*)
ENDFOR
mean ←  tot / howmany                           (* mean of actual scores*)
PRINT "mean score" mean  "for" howmany "simulations"
Diff ← (ABS ( mean – 3.5)
PRINT "difference from expected mean of 3.50 is " Diff
tot ←  0
FOR K ← 1 TO 6
        ADD dice[K] TO tot     (*add number of scores into total*)
meanct ← tot / 6                                (* mean of score counts *)
PRINT "mean of score counts for " meanct  "score " K
ENDFOR
ENDPROG
```

(9 marks)

Examiner's Guidance Notes

Candidates struggled to see how to use RANDOM() to produce an integer number between 1 and six.

B8.



head ptr member-1 member-2 member-3 end-member

a) The diagram above represents a linked list with a single pointer field. Show the pointers if member-3 was removed from the list

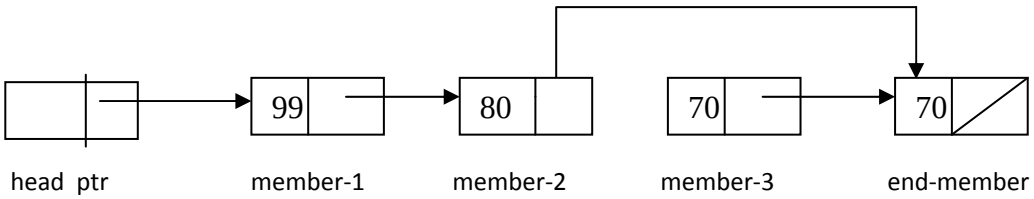
(2 marks)

b) Write code to read in an integer value and delete the first member to contain this value. Note – The input value may not be held in the list.

(10 marks)

Answer Pointers

a)



head ptr member-1 member-2 member-3 end-member

b)

```
PROCEDURE serdelete(VAR headptr  IS  ptr)
LOCAL VARIABLES pos, next  IS  ptr  searching  IS  BOOLEAN  int  IS  INTEGER
PRINT ("input the integer to be deleted from the list")
INPUT (int)
IF headptr NOT EQUAL TO NIL THEN
BEGIN                                           # not an empty list
  IF headptr^.data = int THEN
   BEGIN                                        # first entry in the list
        pos  ← headptr
        headptr ← headptr^.next
```

```
        DISPOSE(pos)
        PRINT int, "deleted from linked list"
    END
  ELSE
    BEGIN
      searching ← TRUE
      pos ← headptr
      next ← headptr^.next
      WHILE (pos^.next NOT EQUAL TO NIL) AND searching DO
      BEGIN
        IF next^.data = int THEN
        BEGIN                              # remove the entry
          pos^.next ← next^.next
          DISPOSE(next)
          searching ← FALSE
          PRINT int, "deleted from linked list"
        END
        ELSE
        BEGIN                              # go to next entry
          pos ← next
          next ← next^.next
        END
      END
    END
END                                                    (10 marks)
```

Examiner's Guidance Notes

Unfortunately most candidates gave only the answer to a), worth only 2 marks.

b) was usually answered by memorized code which they tried to modify. This led to time wasted setting up the list, which was not asked for. The 'delete' part was usually correct, though it rarely deleted the number read in.

B9. One of the standard data structures used in programming is the *queue* with its distinctive operations of *join* (add a new value to the back of the *queue*) and *serve* (remove a value from the front of the *queue*). Using an array as the basic storage for the *queue*, write code in a language of your choice to implement the operations *join* and *serve*.

(12 marks)

Answer Pointers

Basic solution

```
int queue[100]; /*The queue itself */
int joinp=0; /* the joining index*/
int servep=0; /* the serving index*/

void join(int e){
        queue[joinp]=e;
        joinp++;
}
```

```
int serve(){
        int s=queue[servep];
        servep++;
}
```

This can be enhanced by error checks to see if the queue is full before a join and to see if the queue is empty before serve. This can conveniently be handled by having a count variable which holds the number of elements in the queue [add count++ to join() and count-- to serve(), then can have error checks like if(count<=0)EMPTY and if(count>=100)FULL]

The queue can be made 'circular' by reusing the beginning of the array when servep/joinp get to the end, e.g.
if(joinp==100)joinp=0;

Examiner's Guidance Notes

Not a popular question - mostly poor answers

Question specifically asks for code so a description, or diagrams not acceptable

Code full of the word 'stack' not acceptable

Why are the queues so small? - often array was only length 4

There should be no inputting of data

There should be no loops

There was no request to write a full program

There was confusion in code between the indexes of the queue and the values in the queue

Did not ask for circular queue but some students had obviously met the idea - even if they did not quite program it correctly

It is NOT a good idea to serve the queue by shuffling all the elements down one place - even though that is what happens in a people queue

B10. Give one reason why someone might prefer:

a) an interpreter *rather than* a compiler
b) a WIMP operating system *rather than* a command line operating system
c) an object-oriented programming language *rather than* a procedural language
d) a direct access file *rather than* a sequential access file
e) a high-level language *rather than* a low-level language
f) a linked list *rather than* an array

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

(6 x 2 marks)

Answer Pointers

The following gives example answers for each part (other answers are possible)

a) An interpreter can help find run-time errors.
b) Users don't have to memorize commands in a WIMP operating system.
c) An object-oriented programming language increases the software reusability.
d) A direct access file enables fast access to the required record.
e) A high-level language allows the user to concentrate on solving the problem (not the limitations of the actual computer).
f) A linked list enables *space to be allocated when it is required.*

Examiner's Guidance Notes

A very popular question - mostly poor answers

Some candidates wrote whatever they knew about both terms in each part. They were supposed to write one advantage of the given term.

The following are some common WRONG answers;

a) WRONG: An interpreter is easier to use.
b) WRONG: A WIMP operating system is faster than a command line operating system.
c) WRONG: An object-oriented programming language is easier than a procedural language.
d) WRONG: A direct access file is more secure.
e) WRONG: A high-level language is faster than a low-level language.
f) WRONG: A linked list is more user friendly or faster than an array.

B11. When designing a web page to take information from a user there are various form elements available. Name FOUR different form elements, giving an example of the kind of information that each is best suited for and drawing what they look like on screen.

(4 x 3 marks)

Answer Pointers

Names - any four from...

single/multi-line text
menu selection
radio buttons
check boxes
submit buttons
file upload button (legal but rarely offered)

Examples

single text - Name, email, tel no, etc
multi-line text - address
menu selection - 1-from-many or many-from-many choice - e.g. select country
radio buttons - 1-from-few
check boxes - many-from-few
submit button - send info to server

Drawings

accept any reasonable attempt to draw as seen on web browser display

Examiner's Guidance Notes

Many good answers scoring full marks but some poor answers hopelessly off track

This was NOT a question about survey design, so answers which offered survey methods and types of questions (interview, questionnaires, multiple choice, yes/no) were not accepted.
The item variously called Listbox/dropdown menu/select was regarded as one item and was only allowed to be an answer for one part

This was NOT a question about general webpage design and so it should not have been necessary to talk about top bar, left bar, footer bar, etc.

Among the answers that were along the right track it was disappointing to see radio buttons used where multiple choice was obviously possible and check boxes where only a single choice was possible.

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.

a) identifier not declared
b) type error - index not allowed
c) array index out of bounds
d) syntax error
e) variable required
f) type error - invalid type

line	Version 1 (C)	Version 2 (Pascal)
1	void main();	PROGRAM p;
2	int x, y;	VAR x, y : integer;
3	char z[256];	z : ARRAY[0..255] OF char;
4	{	BEGIN
5	w = 1;	w := 1;
6	x = 'w';	x := 'w';
7	'w' = y;	'w' := y;
8	x = z;	x := z;
9	if (b > 1 b = b - 1 ;	IF b > 1 b := b - 1 ;
10	x[0] = y;	x[0] := y;
11	x = 0;	x := 0;
12	while(x < 9) {	WHILE x < 9 DO BEGIN
13	z[x-1] = x / y;	z[x-1] := x / y;
14	x++;	x := x + 1
15	}	END
16	}	END.

(6 x 2 marks)

Answer Points

a) Line 5	identifier not declared	x,y,z declared lines 2,3 but not w
b) line 10	type error - index not allowed	in x[0], x is not declared as an array
c) line 13	array index out of bounds	1st time in loop z[x-1] is z[-1]; -1 is bad subscript
d) line 9	syntax error	need closing); if (b > 1) b = ...
e) line 7	'w'=	cannot assign to a constant
f) line 13	type error - invalid type	z[x-1]= cannot receive float (needs char)

the error “array index out of bounds” occurs on Line 13 not Line 3.