The BCS Professional Examination
Certificate

April 2005

Examiners' Report

Software Development

General

This paper frequently involves developing code logically from first principles. While memorised code may afford speed in writing some answers, in others there is invariably a difference between the memorised terms and those of the question, which leads to difficulty. It also leads to inclusion of additional material not asked in the question. The examiners do not penalise this directly as they consider the time wasted in writing it out as a sufficient penalty.

It is very useful that candidates are given practice in answering questions on paper. Developing programs interactively at a PC or terminal is a poor preparation for this examination. While such practical work is useful in developing students' understanding of a programming language it is NOT good for developing algorithms, or developing a part of a much larger program.

Flowcharts are inferior to pseudo-code for developing algorithms. They consume much time in drawing, imply 'GO TO' instructions and are difficult to develop logically. They become unwieldy for a problem of significant size or complexity.

No marks are given for copying out the question, nor for repeating declarations given in part a, say, in the latter part of the question. Algorithmic development need only show those statements or sections which are developed further, provided that it is clear where they belong, rather than copying all of it out again.

Question 1

1. The file '*stock*' contains details of books currently held in stock by a retailing bookshop:

<ISBN>	<Author>	<title>	<stock level>	<price>
10-digit integer	20 characters	40 characters	4-digit integer	real number
0 7493 97543	Bernieres, L.	Capt. Corelli's mandolin	25	7.99

Another file '*supply'* contains details of books held in a warehouse:

<supplier name>	<ISBN>	<available>	<stock level>	<price>
20 characters	10-digit integer	1 char, T/F	4-digit integer	real number
Vintage Fiction	0 7493 97543	T	650	6.55

a) Specify suitable data structures to hold details for both of these files. Hence write a suitable file description of '*stock*' and '*supply*'. **(6 marks)**

b) The <Author> and <title> fields both use many characters and for a typical file with 1000 books would result in waste of storage space, as much of the available space would not be used. Suggest how this storage space could be minimised without losing processing capability. **(4 marks)**

c) Write an initial algorithm to read the '*stock*' file once and for each entry search the '*supply*' file to match the ISBN of a book. If the book is found in the '*supply'* file and the price lower than on the shop '*stock*' file, print the full details of the book. Count how many records are on the '*stock*' file, how many matches are found and print the results at the end.

An initial algorithm and at least one stage of development must be shown. **(20 marks)**

Answer Pointers

a)

Possible data structures are

```
TYPE stocktype = RECORD
        isbn : longint
        author : array [1..20] of CHAR;
        title : array [1..40] of CHAR;
        stocklevel : shortint;
        price : REAL;
        END;

TYPE supplytype = RECORD
        isbn : longint;
        supplier : array [1..20] of CHAR;
        available : BOOLEAN;
        stocklevel : shortint;
        price : REAL;
        END;
```

b)

There are several possibilities. Some procedural languages (like FORTRAN) allow dynamic declaration of arrays, so the actual size is not fixed and only storage space needed is used. A linked list could be used here in PASCAL so that much space is saved where there are short titles and the occasional very long title won't cause an overflow. It would also be possible to use temporary files.

c)

Initial algorithm:

```
assign files
set counters to zero
WHILE NOT end-of-file on stock file
    READ  stockfile record
    increment stock record count
        WHILE NOT end-of-file  on supplier file
        BEGIN
        READ suppler record
             test for match of isbn  from both files:
        IF match  THEN BEGIN
            increment match-count
            IF {available AND lower price} THEN PRINT record details
            END
        END
Final counter output
END.
```

Development:

```
ASSIGN stock, supply files, open for reading {always system dependent}
rec_ct ←0      match_ct ← 0
WHILE   NOT   EOF(stock) DO
      BEGIN
          READ stock, stock_item
            rec_ct ← rec_ct + 1
      match ← FALSE
      WHILE   NOT EOF(supply) AND (NOT match) DO
            BEGIN
            READ(supply, supply_item)
            IF stock.isbn  =  supply.isbn THEN
                  BEGIN
                  match_ct  ← match_ct + 1
                  match ←TRUE
                  IF {item is available} AND supply.price < stock.price
                      THEN {PRINT book details}
                  END
            END  {inner while loop }
      RESET supply-file
          END  {outer while loop }
WRITE number of records on stock file, rec_ct
WRITE number of matches, match_ct
CLOSE supply file, stock file
END
```

Examiners' Comments
Marks were given for correct structure (however shown) as well as appropriate instructions. No development at all was penalised, as this was specifically required by the question.

Students should be able to choose the correct type of loop for a particular problem. Thus a 'FOR' or count-controlled loop is inappropriate here. The initial algorithm should have been checked before developing the code and any errors or omissions corrected.

Excessive use of simple procedures containing one or two instructions are to be avoided.

A wide range of ability was shown in part c). Mostly students had some idea of the correct program structure; marks were given for correct components with more for correctly-terminated loops. Only a few wasted time by repeating declarations made in part (a). About half of the answers showed little or no development of the initial algorithm, for which 'added-value' marks were awarded. If there was only one algorithm in a developed (even coded) stage marks were given for every valid instruction and loop.

Question 2

2.
```
line      Program best-of-three
number    main variables a, b, c : INTEGER
1         Procedure select (t1,t2,t3)
          BEGIN
2         If t2 > t1 THEN swap (t2, t1)
3         If t3 > t2 THEN swap (t3, t2)
4         If t2 > t1 THEN swap (t2, t1)
          END
5         INPUT (a, b, c)
6         select (a, b, c)
7         WRITE (a, b, c)
          END
```

a) Dry run the algorithm with input values 1 3 2 at the top level. Use the line numbers in your answer. **(12 marks)**

b) Certain languages (e.g. Pascal) do not incorporate a system procedure ‘swap’. Write appropriate code for this procedure and state where it would be placed in the supplied code, with reasons. **(7 marks)**

c) What property must the variables used for procedure parameters (t1, t2, t3) have for the desired result to be achieved? **(4 marks)**

d) In some languages the variables (a, b, c, t1, t2, t3) can be strings of characters. Where could this be applied? What constraints are needed to make this work properly? **(7 marks)**

Answer Pointers

a)

line number /instr	**test**	**t1**	**t2**	**t3**	**a**	**b**	**c**	**output**
5 / Input	?	?	?	?	1	3	2	
6/ procedure call	?	?	?	?				
1 / procedure code	?	1	3	2				
2 / IF t2 > t1	true	3	1	2				
3 / IF t3 > t2	true	3	2	1				
4 / IF t2 > t1	false	3	2	1				
End procedure code								
6 / procedure return					3	2	1	
7/ Write								3 2 1
end								

b)
```
Procedure swap (x,y)
local temp : INTEGER
BEGIN
    temp ← x
    x ← y
    y ← temp
END
```

‘Procedure *swap*’ would be placed above the ‘procedure *select*’ code in those languages which require procedures to be placed before the code that invokes them. Thus ‘procedure select’ is placed before the top level which invokes it.

c)
The variables must be capable of passing back the changed values held in them after the code has been run as well as initially receiving initial values from the top level variables. Hence invoke : a $\rightarrow$ t1, b $\rightarrow$ t2, c $\rightarrow$ t3 and on return : t1$\rightarrow$ a,
t2 $\rightarrow$ b, t3 $\rightarrow$ c.

d)
This could be used in word-processing code to exchange characters in written documents. Problems would be the matching up of the strings of characters, particularly if these are of variable length. A constraint as to the maximum length of such characters would have to be found, such that space was not wasted by having the same length as the largest string for all the character variables. Much more code would be needed especially if blocks of such characters are swapped.

Question 3

3. *a)* Write a procedure, in Pseudocode, or Structured English or a programming language with which you are familiar, that implements **insertion** in an array of data ordered in an ascending manner as follows:

The procedure should accept an integer parameter EL that is the data to be inserted.
The procedure should scan an array (name = LIST, size = 100).
The first element in the list contains the number of data elements it contains. That is, LIST[1] = number of elements in LIST.
All existing data in LIST is held together, that is all data is stored adjacent, ascending.
On finding where to insert the new data, you will need to make a space and move the existing data to create an empty space.
If LIST is full, that is, there is no space to insert the new value, an error message should be given. If successful, no return message need be given. **(20 marks)**

b) Dry-run your pseudocode from *a)* above with the following data:

EL = 23

LIST =

4	10	15	22	24					
1	2	3	4	5	6		98	99	100

(10 marks)

Answer Pointers

a)

The procedure should accept an integer parameter EL that is the data to be inserted.

The procedure should scan an array (name = LIST, size = 100).

The first element in the list contains the number of data elements it contains. That is, LIST[1] = number of elements in LIST.

All existing data in LIST is held together, that is all data is stored adjacent, ascending.

On finding where to insert the new data, you will need to make a space and move the existing data to create an empty space.

If LIST is full, so that there is no space to insert, an error message should be given. If successful, no return message need be given.

Notes: The array has subscripts 1 to 100, not 0 to 100, or 0 to 99.
The maximum number of ascending data numbers that can be stored is 99.
When requiring the index of the last used position in the array it is LIST[1]+1.

Procedure INSERT should

- Accept EL as parameter, and prepare to insert.
- Check list not full, LIST[1]=99. If so, return error message.
- If list is empty, LIST[1]=0, then LIST[2]:= EL, increment LIST[1] and exit.
- Scan for insertion point. Scan with index, say i, from 2 until LIST[1], stopping when EL<LIST[i] or we exhaust the scan range. The insertion position is now i.
- Shift elements LIST[i+1] to LIST[(LIST[1]+1)] up one place. A loop, say with index j set to run **down** from (LIST[1]+1) to i, moving each LIST[j] into LIST[j+1], and decrementing j until j=i, the insertion point.
- Now insert EL, LIST[i]:=EL
- Increment LIST[1]

b) Dry run should show something like:

EL	LIST[1]	i	j	LIST[i]	LIST[j]	LIST[j+1]
23	4	2 until 5	6	10		
		3		15		
		4		22		
		5. Stops because EL<24. i is insertion point.		24		
			5 until 5		j runs (4+1) down to 5	
		5	5	22	24	24
					23	
	5					

5	10	15	22	23	24				
1	2	3	4	5	6		98	99	100

Examiners' Comments
The simple approach given in the solution for part (a) is sufficient. There are more complicated algorithms that would be acceptable but they would cost candidates valuable time to write down. For example finding the correct position by binary chop or adding the new number at the end and then coding a complete sort of the array.

There is a nice, simple algorithm that starts at the end of the array (position 5 in the example) and works backwards moving elements one place up until the place for the new element is found and then the new element is just added in the last created 'hole'.

The "list full" test was often written LIST[1] equals 100. In fact the LIST is full when LIST[1] equals 99.

Many answers did not work if the list was empty initially so there should have been special code to test for an empty list and insert EL as the new (only) data element.

The question asks for a procedure with parameters. The procedure is to work on a LIST which already contains some data. Thus no answer should have had any input or output, reading or writing.

The data in LIST was given and was already sorted. Thus no answer should have started by sorting the LIST.

No answer should have had any pointers or records.

The names LIST, EL were given in the question so no answer should have called these items by any other name.

The code to move a sequence of elements one place further up the list must start with the highest numbered subscript and work down to the lowest. If the move is done from the lowest subscript upwards, the result is that the 'moved' elements will all be the same value (the first one 'moved')

Question 4

4. *a)* Algorithmic design in Structured Programming uses three principles. Describe EACH of these principles. **(12 marks)**

b) Give an example of each principle applied to **CODE.** **(9 marks)**

c) Give an example of each principle applied to **DATA.** **(9 marks)**

Answer Pointers

a) Structured programming uses principles
- Sequence,
- Selection and
- Iteration.

b) Sequence is code is begin..end block, or similar in other programming languages.

Selection in code is if..then..else.. , or similar in other programming languages.

Iteration in code is a loop.

c) Sequence in DATA is a record or structure, or similar in other programming languages.

Selection in data is a variant record, or union or alternate structure, or similar in other programming languages.

Iteration in DATA is an array or file.

Many candidates answered parts (a) and (b) and either repeated their answer to part (b) in part (c) or left part (c) blank so 9 marks were lost immediately. Data structure diagrams are a convenient way of showing sequencing, selection and repetition in data

```
      +---+           +---+       +---+
   |seq|           |sel|       |rep|
   +---+           +---+       +---+
    / \             / \          |
+---+ +---+     +---+ +---+    +---+
|   | |   |     |  o| |  o|    |  *|
|   | |   |     |   | |   |    |   |
+---+ +---+     +---+ +---+    +---+
```

Examiners' Comments
Candidates seemed able to describe selection and repetition but were unable to find any words to describe sequencing. However it was not sufficient to say that "selection means selecting..." or that "repetition means repeating...". When describing a term T you have to find words different to T.

In section (b) many candidates gave very elaborate examples, often complete programs. For selection something simple like "if x>y then x:=y" or "if(x>y)x=y;" would have been enough.

In many answers the examples requested in part (b) were actually written in part (a). Strictly speaking the marks for the examples should not have been awarded unless they were written in the correct section, but in this case the examiner was lenient.

Some unsatisfactory answers gave a definition of the Software Development Life Cycle or a description of an algorithm.

Section B

Question 5

5. Briefly describe the type of testing you would use for:
a) Unit tests **(6 marks)**
b) Integration testing **(6 marks)**

Give reasons for your answers.

a) For unit tests, use logic tests ('white' box testing where the code is available). Test cases are designed to force toe algorithm to take all the coded paths. Sometimes a series of tests are necessary. Output for each test must be checked against known or calculated results.

3 marks for white box testing with appropriate definition and **3 marks** for further material, particularly about how it is done with reasons, e.g. this type of testing verifies that individual units function correctly and are appropriately coded.

b) 'Integration is testing the assembled units, and checking that interfaces generate and accept correct data with fail messages being produced for incorrect data. this is usually termed 'black' box testing, where the code is not available. Black box tests may necessitate further white box tests of units.
3 marks for 'black box' or 'functional' testing; **3 marks** for description about how it is done with reasons.

Examiners' Comments
A bookwork question answered by nearly all candidates. Generally done quite well with some good answers. Others confused 'black' and 'white' box testing, or used both forms everywhere. Others wrote everything they knew about testing from memory and hoped the examiner would pick out what was relevant. Some candidates expressed their answer in bullet-point form, which is to be commended. Repetition of points does not gain extra marks. Also students should avoid using the word 'integration' in the definition of integration testing. ('Integration testing is where the modules are integrated and tested.')

Some students wrote quite brief answers and made few points beyond a basic definition like 'unit testing is where the modules are tested separately'. Students need practice in answering this kind of question, and to realise that more than that is expected for 6 marks!

Students must realise that at this level unselective regurgitation of memorised notes will attract fewer marks than answers which supply relevant material. Simple logic would help here and avoid errors in recall like 'Unit testing needs black box testing and integration testing needs white box (or is it the other way round)? Thus 'integration testing is for the whole system (parts put together) hence impractical to do white box testing which looks at logic and actual code - there would be a lot to do! Hence use black box testing, which looks at interfacing and input/output.'

Question 6

6. An inter-island ferry has a passenger-carrying capacity of C people. In general W people are waiting for the ferry. If W > C then those left behind are P percent of those waiting and L percent of the ferry's capacity.

a) Derive separate expressions for P and L in terms of the variables (C, W). Simplify these expressions as far as possible. **(6 marks)**

b) Write a program
i) to accept a value for C
ii) to accept lower and upper limits for W
iii) to accept a value for W and
iv) then print a table with the headings below

People waiting (W)	Number of people left behind	% of people waiting left behind (P)	% of ferry's (L) capacity left behind

(6 marks)

Answer Pointers

a)

$\frac{W - C}{W} = P$ - fraction of people waiting left behind (if C > W this is zero)

$\frac{W - C}{C} = L$ - fraction of people left to capacity of ferry.

$P = 100 \left(\frac{W - C}{W} \right) = 100 (1 - C / W)$

$L = 100 \left(\frac{W - C}{C} \right) = 100 (W / C - 1)$

} these equations express P and L in terms of C and W

Substituting for C we get $P = 100\left(\frac{L}{100 + L}\right)$

Substituting for W we get $L = 100 (W / C - 1)$

b)

<u>ferry program</u>

```
100 PRINT "ferry passenger capacity program"
110 PRINT "Implemented in Qbasic"
120 PRINT "input ship passenger capacity/legal safety limit"
130 INPUT C
140 PRINT "input LOWER limit for number of people waiting for ferry"
150 INPUT low
160 PRINT "input UPPER limit"
170 INPUT high
180 PRINT "total waiting  left behind  percent     capacity percent"
185 REM 'FOR' loop to print table between input limits
210 FOR W = low TO high STEP 5
215 IF C > W THEN LET B = 0 ELSE LET B = W - C
220 IF C > W THEN LET L = 100 * W / C ELSE LET L = 100 * (W / C - 1)
230 LET P = 100 * L / (100 + L)
240 PRINT TAB(5); W; TAB(17); B; TAB(30);
250 PRINT USING "###.#"; P; TAB(45);
260 PRINT USING "###.##"; L;
300 NEXT W
350 END
```

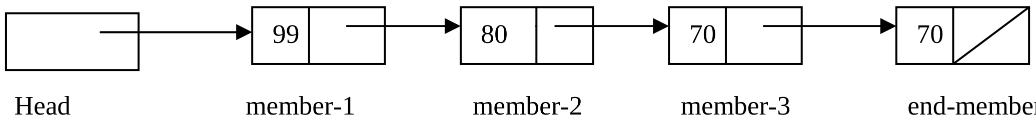
Examiners' Comments

a) required explicit equations for P and L in terms of input values W and C. Some candidates did not realise this and had a page of algebra; others managed it quite easily.

b) wanted a <u>table</u> of values for W and corresponding P, L values. Thus a loop between input maximum and minimum values for W was needed with (W -C), P and L values calculated from the equations derived in (a) was necessary**.** Incorrect equations derived in (a) were not penalised in (b). To get all the marks proper captions such as “input the number of people waiting for the ferry” rather than “input W” were expected. Again answers ranged from completely correct to the bizarre.

Question 7

7. The diagram below represents a linked list with a single pointer field.

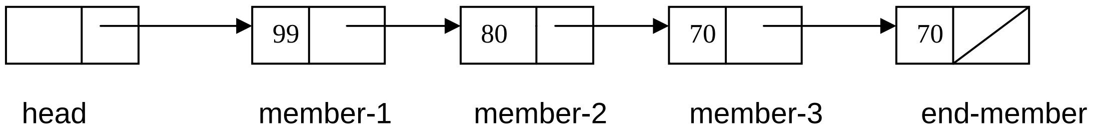


a) Write a declaration for one member of this list in a suitable language. State which language you are using. **(3 marks)**

b) Copy the diagram and show the necessary pointer movements to exchange member-2 with member-3. **(5 marks)**

c) Write code for these pointer movements in the same language you used in part *a)*. **(4 marks)**

Answer Pointers

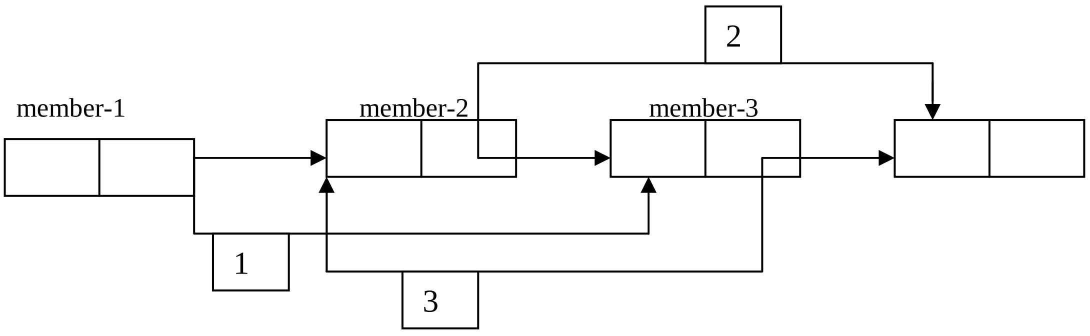


The diagram represents a linked list with a single pointer field.

a)

```
TYPE     ptr   =  ^ node;                    PASCAL used
         node = RECORD
         data : INTEGER;
         next : ptr
            END;
```

b)



c)

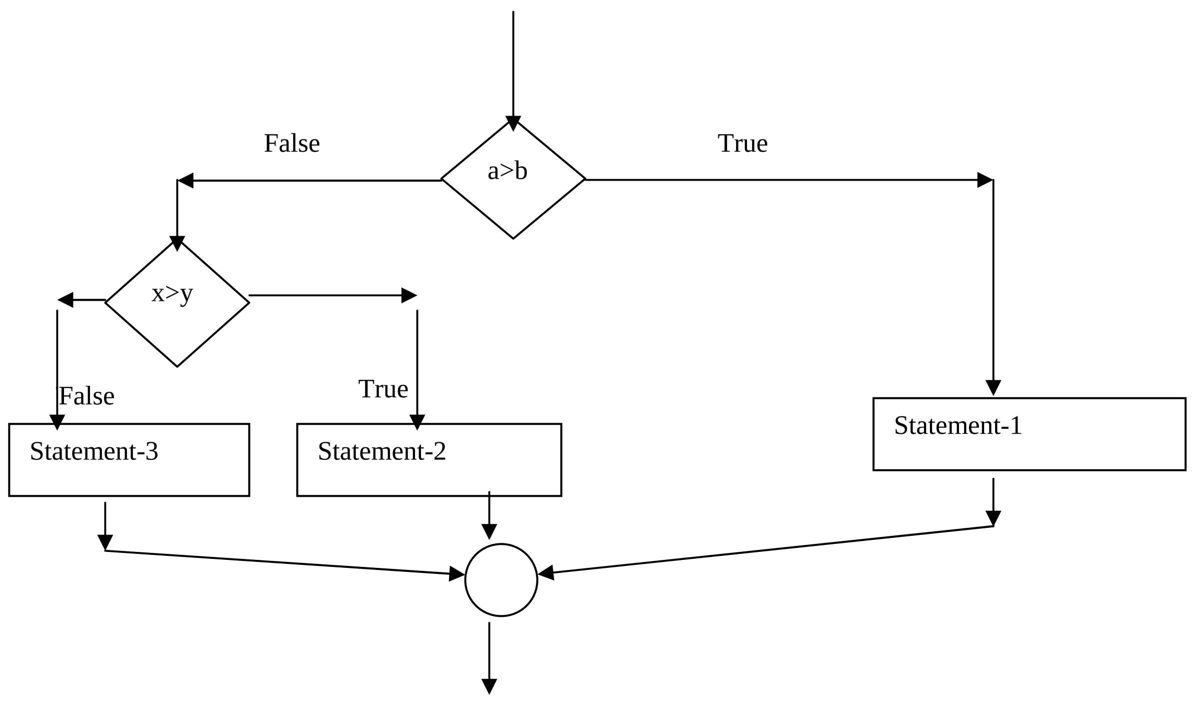
```
temp := member1↑.next                      {address needed in stage [3]}
member1↑.next := member2↑.next             {stage  [1] }
member2↑.next := member3↑.next             {stage [2]}
member3↑.next := temp                      {stage [3]}
```

Examiners' Comments

It is essential that the pointer movements are made in the correct order, otherwise the address in one field altered too early is wiped out before it is needed in a later phase.
To exchange these members within a linked list it is obviously necessary to traverse the list until member-1, member-2 and member-3 are in the right places.

Question 8

8.



a) Represent the flowchart (above) in pseudocode. **(4 marks)**

b) Represent the pseudocode below by a flowchart.

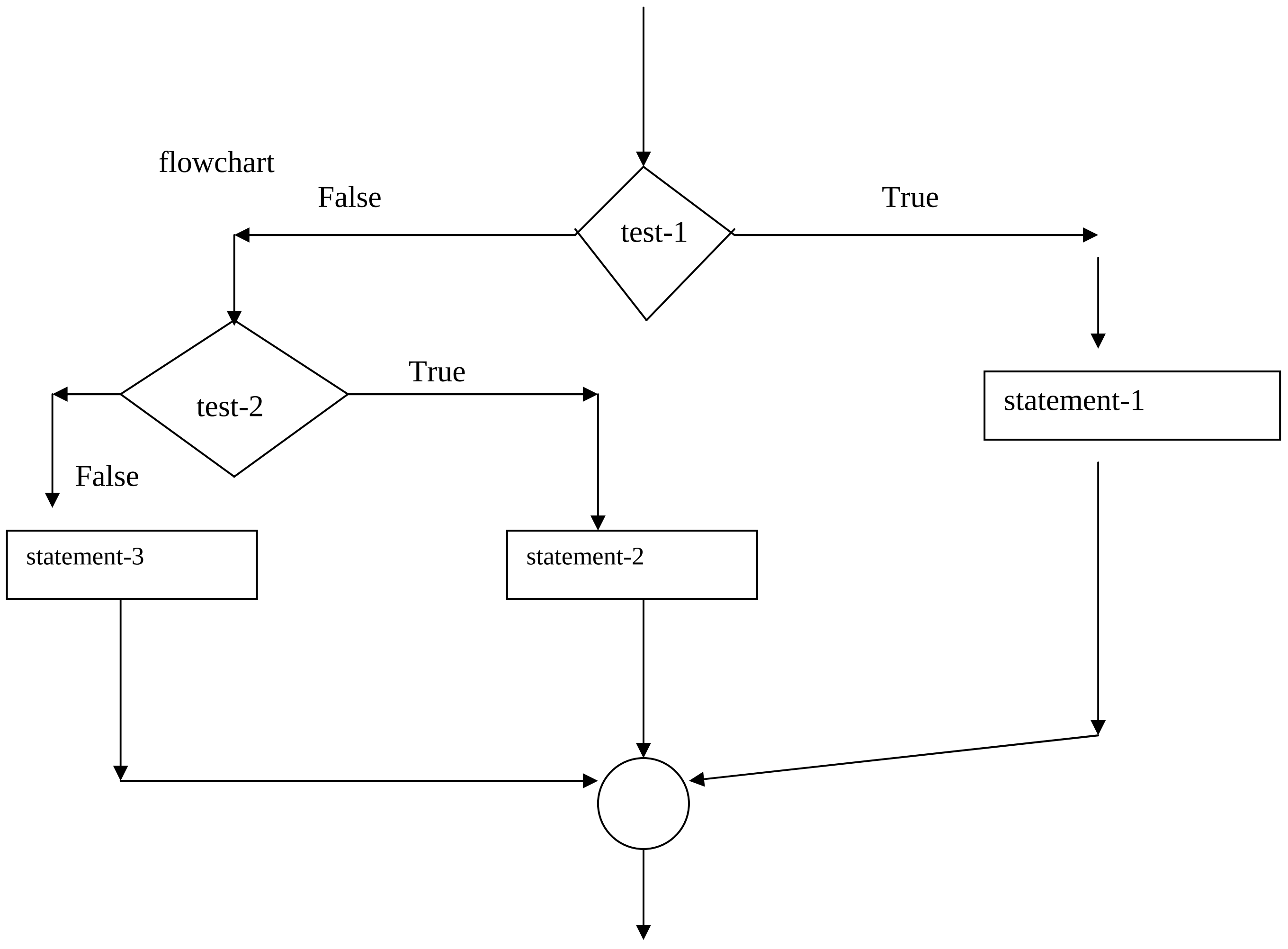
```
IF test-1 THEN          IF test-2     THEN statement-1
                                      ELSE statement-2
         ELSE statement-3
```

(4 marks)

c) Specify appropriate test data for *b)* so that each of the three statements is output. State which output should be produced for each set of test data. **(4 marks)**

Answer Pointers

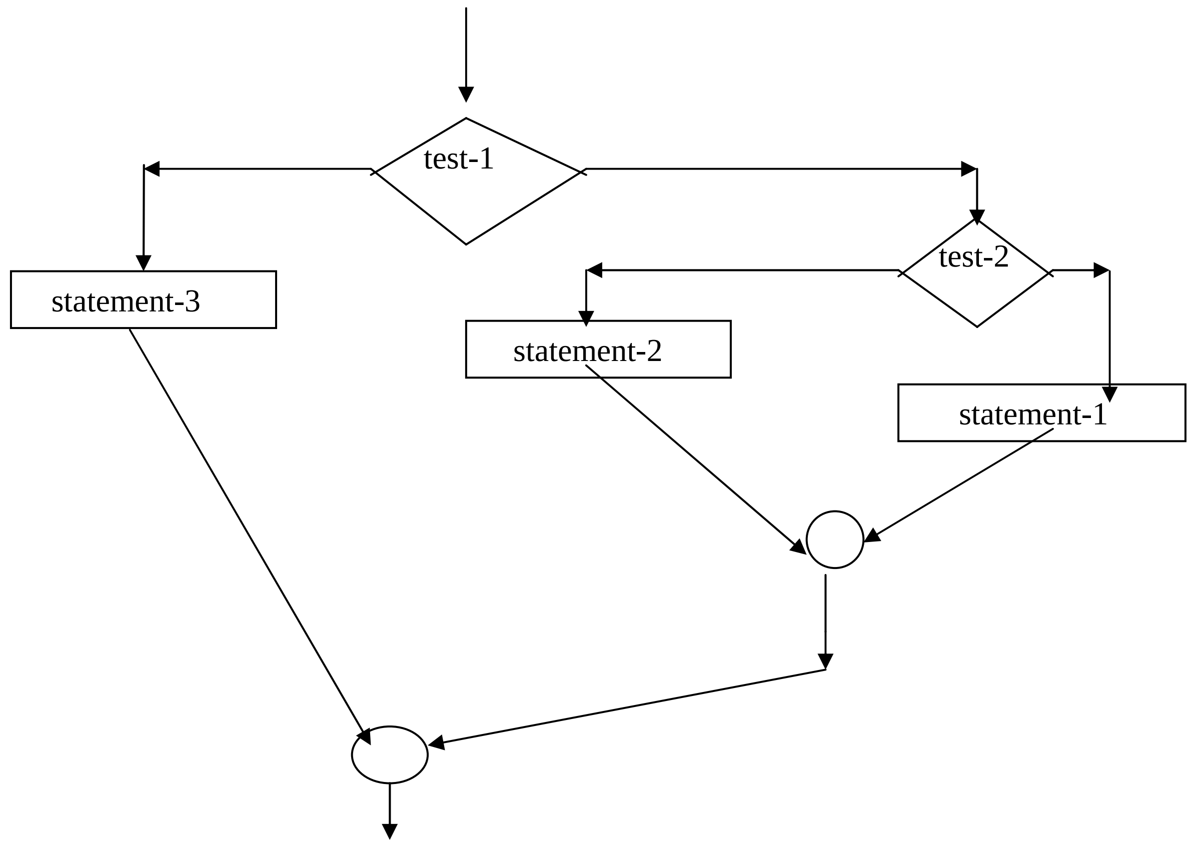
a)



a) **IF test-1 THEN statement -1**

```
IF test-1   THEN statement -1
            ELSE IF test-2   THEN statement-3
                             ELSE statement-3
```

b)



c)

Only the simple truth table given below is wanted here.

test-1	test-2	output/execution
true	true	statement-1
true	false	statement-2
false	(true OR false)	statement-3

Question 9

9. *a)* Describe one purpose of an index-access file? Give reasons for your answer. **(6 marks)**

b) When in operation, is it better to keep the index in main memory, or on disk? Give reasons for your answer. **(6 marks)**

Answer Pointers

a) An index on a file is for speed of access. Its purpose is to respond to random requests for records such as from a telephone query.

b) Keeping the index in memory makes for faster access because electronic speed are higher than disk-based electro-mechanical speeds.
Keeping index on disk retains integrity in event of system crash.
Either conclusion acceptable so long as it is supported.

Examiners' Comments

Many answers for part (a) did not score good marks because the answer was a *description* of an index-access file rather than a *purpose* of having an index-access file. So detail about how the index is created, how it is updated, etc did not score marks because it did not tell the purpose of having an index.

In both parts of this question many candidates used the word "easier" in a way that was not accepted. Using an index on a file is not "easier". It is harder. There is more work to do in making the index and keeping it up to date. Accessing a record using an index is not "easier". The "easiest" thing to do is start at the beginning and look through all the records until you find the one that you want. The benefit that you get with an index is speed, you can find records "faster".

Candidates need to be careful with terminology. There is a big difference between "looking for a file" and "looking for a record".

When giving examples it is not enough to say something like "An example of an index-access file is a telephone bill". A batch production of telephone bills could work well on a sequential file. The example that requires index-access is responding to a telephone enquiry which requires *random* access to the customer database in order to be answered quickly.

Something about the way part (b) of this question was worded confused many candidates. The question in the sentence is not "When ... ?" but "is it ... ?"

Note that a diagram like

```
+---+---+---+---+
| a | c | d | j |
+---+---+---+---+
```

is not acceptable as a diagram of an index. It is necessary to have key/address pairs.

Question 10

10. Show a GUI design for a system that is used to monitor 4 different locations using four remote cameras. Include in your design GUI elements of a drop-down list and a set of radio buttons, and give them appropriate functions to perform. Explain your design fully. **(12 marks)**

Answer Pointers

Expect a screen with four windows. The drop-down list should be for some function that is itself a rather long sequence. This might be a degree of camera movement (up-down, left-right), or similar. The radio-button should be for some data that represents a choice from alternatives. This might be brightness, or a zoom ratio.

Examiners' Comments
This question asked for a DESIGN, so there were no marks for an answer which simply described the purpose of the various interface elements. The question asked for the design to be EXPLAINED fully, so a diagram on its own could not obtain full marks.

It was not acceptable to have the drop-down list and the radio buttons used for the same purpose e.g. both allowing the selection of one camera out of 4.

It was clear that some candidates were visualising 16 cameras with 4 in each location.

In a drawing/diagram only one entry in a drop-down list is visible so it is vital that the other entries are shown or described in the explanation.

Among the answers there were designs which involved choosing a camera on one page, then choosing a zoom on another page and then a brightness on another page, etc. These were not considered good designs as it seemed better to have all the control situated on a single page.

Question 11

11. Describe the functions of the following software elements. State whether each is 'system software' or 'application software'.

a) A Compiler **(4 marks)**
b) A Scheduler **(4 marks)**
c) A Word Processor **(4 marks)**

Answer Pointers

a) Compiler is system, and translates source to executable.
b) A scheduler is system, and sequences tasks for execution on the CPU.
c) A Word Processor is Application, and helps users to format text and diagrams in a document.

Examiners' Comments
It was not enough to say that a compiler compiles things, that a scheduler schedules things, a word processor does word processing, etc.

For part (b) some candidates gave answers relating to "cron jobs" or similar and some candidates gave answers relating to "diary" type programs.

Several answers focussed on relatively unimportant, secondary features of the software concerned. It is not correct to claim that the principle task of a compiler is to find syntax errors in a program. It is not correct to claim that the principle task of a word processor is to make line drawings.

Question 12

12. Briefly describe the type of documentation you would assemble for the maintenance of a desktop database application. Give reasons for your answer. **(12 marks)**

Answer Pointers
For maintenance, need to document the internal design, workings and tests for the functions so that changes can be made.

Also, the documentation should include design relating to the database (ERD or SQL), Test specifications, and Test results.

Also, the documentation should include details of how the application was generated - version number and date**.**

Examiners' Comments
This question was quite poorly answered in general.

It was not sufficient just to describe documentation in general. It was not acceptable to answer the question "Why produce documentation?". It was not acceptable to answer the question "How is a software product documented?".

The question asked about a specific kind of application. Full marks could not be obtained unless the answer was written specifically for MAINTENANCE and specifically for a DESKTOP DATABASE application.