The BCS Professional Examinations
Certificate

October 2005

Examiners' Report

Software Development

General Comments

Programming questions are designed to test skill in solving problems, rather than tours-de-force of memory. (Other examiners call it 'Brain Dump'.) Practice for this is inculcated by regular marked homework rather than practical work in a computing laboratory. The Software Development paper does not ask questions about the syntax or use of a specific programming language (like C) because of the wide diversity of languages studied by the candidates world-wide. Lengthy practical sessions merely heighten the importance attributed to syntactical correctness in a particular language by students, and downgrade algorithmic development, as numerous runs of the program on a PC take its place.

Nearly all candidates' answers to programming questions include system directives like # *include <iostream.h>* or *void main()* which are not required in an examination answer. Moreover, their common occurrence suggests that the candidates have little idea what these directives do.

No marks are gained by copying out the question into the answer book. This is not penalized, but it wastes valuable time. No extra marks are gained by repeating material from one part in a later part. Candidates should indicate where such declarations belong in code written later. The Examiner usually instructs the candidates not to do this in any case.

Question 1

1. The code below affords an approximate value of the square root of a real number (A) from a given initial value (Y)

```
For inner = 1 TO 3 DO
BEGIN
  new = (Y + A / Y) / 2
  Y = new
END
```

a) Incorporate this code into a fully-coded function *sqroot*() with appropriate parameters. Modify the supplied code so that the iterations cease when successive values differ by less than 0.0005. Use A/2 for the initial value of Y. Incorporate a count of the iterations. Use meaningful names for the identifiers. State which language you use. **(15 marks)**

b) Write a test harness, which prints a table of square root values (in increments of 1) between integer input values 'lower' and 'upper' which are requested interactively, and using procedure *sqroot*. It must also print the corresponding values obtained by using the standard function (e.g. SQRT in Pascal). These values should be printed to four decimal places in a field width of 8 characters. **(15 marks)**

Answer Pointers

a)

```
PROCEDURE sqroot(inroot:INTEGER; VAR result:REAL; VAR ct:INTEGER);
                              CONST Error = 0.0005;
                              VAR old,new,diff,Y:REAL;
                              BEGIN
                              calroot :=  inroot/2;  ct := 0;
                              REPEAT
                                        ct  := ct + 1;
                                        new := (calroot +  inroot /calroot) / 2;
                                        old := calroot;
                                        diff := ABS(old - new);
                                        calroot := new;
                              UNTIL diff < Error;
                              result := new
END;
```

PROGRAM sqroot(INPUT,OUTPUT);

```
{prints table of approx. square roots and those from the standard SQRT() function}
     VAR lower,upper,tab,cycles:INTEGER;  avalue:REAL;
     {procedure for approx. and exact square roots with iteration count GOES HERE}
     BEGIN {top level}
          WRITELN('table of square root values with exact counterparts');
          WRITELN('input LOWER limit'); READ(lower);
          WRITELN('input UPPER limit'); READ(upper);
          WRITELN('number approx.root  exact root  cycles');
          FOR tab := lower TO upper DO
            BEGIN
            sqroot(tab,avalue,cycles);
            WRITELN(tab:5,' ':3,avalue:8:5,' ':3,SQRT(tab):8:5,' ':6,cycles:3);
            END;
          WRITELN('program finished normally')
     END
```

Examiner’s Comments

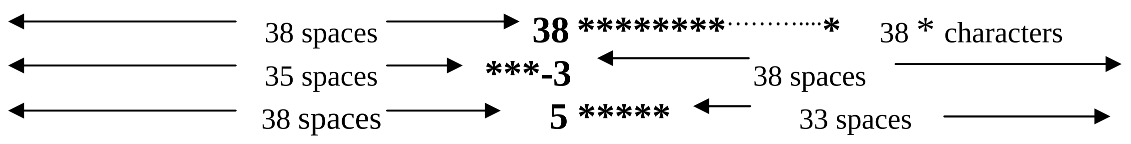
Not popular. A wide range of ability shown, from those who got it completely right to those who merely copied out the given code, sometimes with requested input for the number to find the square root of. Most common omission was failing to print the number of iterations needed to reach the desired accuracy. Just a few used database languages, quite unsuitable for this work.

Question 2

2. Develop a program from the initial algorithm given on the next page. The program reads a sequence of not more than 120 integers terminated by a zero value. A graphical, scaled representation (bar chart) of the data is to be constructed as follows:

Each integer (N) is to take up 4 character positions and is to be printed in the centre of a line 80 characters wide with a row of asterisk (*) characters on the right hand side of N if N is positive and on the left hand side of N if it is negative. The rows of asterisks are scaled such that the largest value of N produces 38 asterisk characters on its row and the other rows contain a proportionately scaled number of asterisk characters. The asterisk characters are to be grouped on either side of the integer (N) as shown in the diagram below:

(actual printed items shown **bold**)



from input data item	1	38
data item	2	-3
data item	3	5
data item	4	0 (terminator)

Initial Algorithm:

```
LOOP until all data input
    READ data item → data structure
    FIND largest data item from data structure → largest
ENDLOOP
COUNT number of data items → ncal
PRINT "number of data items =" ncal
CALCULATE scalefactor → scale
LOOP  to draw diagram
    CALCULATE number of * characters in row  → nchar
    IF data item is positive
      THEN LOOP to print spaces
      PRINT data item
      LOOP to print  nchar * characters
    ELSE  LOOP to print spaces
      LOOP to print nchar * characters
      PRINT data item
    ENDIF
ENDLOOP
```

(30 marks)

Answer Pointers

DEVELOPMENT

Although nine points seems a lot, some of the developments are obvious and students may subsume them into other stages.

***1** A 'while' loop is expected here with terminator as 'data item <> 0'**.**

***2** An array is quite suitable as a data item store; possibly better would be a temporary file. In the final program DATA:ARRAY[1..20] of INTEGER was used.

***3** This is easily incorporated into the input loop which stores in 'largest' the biggest item found so far. Thus we can use READ(data[item]); IF data[item] > largest THEN largest → data[item]

***4** Likewise this is incorporated into the 'while' loop as an 'add 1 to count' instruction.

***5** To calculate the scalefactor so that the largest number input produces the right number of '*' characters to just fill the screen we need to supply the screen half-width as a constant. (Half width, as the format is

<left side for negative data> : <right side for positive data>)

so the scalefactor is width / largest

***6** The loop to draw the diagram is best coded as a ‘for’ loop as the number of input data items is known now. So we can use FOR count = 1 TO ncal; each data item is then data[count].

***7** Following on from *5, the number of ‘nchar’ characters needed for a particular data item is ABS(ROUND(data[count] * scale)) ‘ABS’ is needed as data items can be +ve or -ve. ‘ROUND’ gets the nearest integer after multiplying the data item by the scalefactor.

***8** The loop to print the appropriate number of spaces depends on the language used; procedural languages offer different facilities for ‘PRINT’ or ‘WRITE’ instructions. In Pascal we don’t need a loop as variable-controlled formatting is possible, thus WRITE(' ':width) or better
CONST space = ‘ ‘; … WRITE(space : width).
A loop like `FOR pos := 1 TO nchar DO WRITE('*');`
is used to print the ‘*‘ characters for positive data items.

COMPLETE PROGRAM

```
PROGRAM diagram(INPUT,OUTPUT);
{produces diagram from numerical input BCS A-question}
CONST width = 38;
VAR count, largest,pos,nchar,ncal: INTEGER;  scale:REAL;
    DATA:ARRAY[1..20] OF INTEGER;
BEGIN
  count :=1;  largest := -maxint;
  WRITELN('input first data item'); READ(data[1]);
  {input loop}
  WHILE data[count] <> 0 DO
     BEGIN
     {find largest data item input }
     IF ABS(data[count]) > largest  THEN largest := ABS(data[count]);
     count := count + 1;
     WRITELN('input next data item:end on zero');READ(data[count])
     END;
  {establish how many data items input}
  ncal := count - 1;
  WRITELN(ncal:3, ' data items input');
  {calculate scalefactor}
  scale := width / largest;
  WRITELN('scalefactor = ',scale:6:3);
  WRITELN;
  {loop to draw diagram}
  FOR count := 1 TO ncal DO
     BEGIN
  {get scaled number of * characters}
     nchar := ABS(ROUND(scale*data[count]));
     IF data[count] > 0 THEN
  {write * characters for positive data items}
        BEGIN
        WRITE(' ':width);
        WRITE(data[count]:4);
        FOR pos := 1 TO nchar DO WRITE('*');
        WRITELN
        END
       ELSE
        BEGIN {write * characters for negative data items}
        FOR pos := 1 TO (width - nchar) DO WRITE(' ');
        FOR pos := 1 TO nchar DO WRITE('*');
        WRITELN(data[count]:4)
        END
     END {FOR loop}
END.
```

Examiner's Comments
Moderate popularity and a wide range of marks awarded. Very few started with the given algorithm and developed appropriate points. Most launched straight into code with poor identifier names (x, I and the like). Just a few were quite successful. Many had more loops than required. Others just translated a few ideas or wrote down the undeveloped algorithm. Students need more practice at this kind of work, for development has applications in programming jobs.

Question 3

3. *a)* Write a function, in Pseudocode or Structured English or a programming language with which you are familiar, that implements a search as follows:

The function should accept the target for the search, an integer parameter NUM.
The function should scan an array (name = ASSEMBLY, size = MLEN, both given as global in the context of the function), looking for NUM.
The result of the function should be the index position in ASSEMBLY if NUM is found, or zero if NUM is not found. **(20 marks)**

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

NUM = 11
MLEN = 10
ASSEMBLY =

3	5	7	9	11	13	15	117	21	23

(10 marks)

Answer Pointers

a) The function should
- Accept as **parameter** the NUM, and prepare to deliver the index position in ASSEMBLY
- **Initialise** local variables including some flag or marker that NUM has been found, initially false.
- Set up a single **loop** to scan ASSEMBLY
- The loop should **terminate** if *either*
  - o NUM is found *or*
  - o there are no more subscript positions to try
- When the loop terminates the function **returns** a numeric result

The 20 marks were distributed as follows:

3 for function header, 3 for declarations, 3 for initialisation, 5 for setting up the loop, 3 for the conditional test, 3 for returning the result

b) Dry run should show something like:

Line of code	NUM	i (Loop counter)	MLEN	ASSEMBLY[i]	Found Flag	Result
initialise	11				False	
loop		1	10	3		
loop		2		5		
loop		3		7		
loop		4		9		
loop		5		11	True	5

The function returns 5

The marks were distributed as follows:

2 for tabular presentation, 1 mark for link to code via line number or label, 2 marks for a clear column heading and 2 different correct entries below, 2 marks for another heading & entries below, 1 for discovering the position of NUM

Examiner's Comments
The array could have subscripts 0 to MLEN-1 or 1 to MLEN depending on your chosen programming language.

The question does not state that the array ASSEMBLY has been sorted and the example data in (b) is not sorted so that the binary chop method of sorting is not appropriate.

Unless your command of the English language is good, you are advised to write program code in a recognised programming language rather than using pseudo code.

The function takes some information as a parameter (MLEN) and other information as global (ASSEMBLY) so there should be no reading or input of data in your answer. The function returns a single numeric result so there should be no writing or output of messages in your answer. You should not invent messages like "target found".

The question clearly asks for a function with parameter returning a numeric result. This means that a main program in C is not correct.

The question clearly asks for the *index* of the place where NUM is found to be returned as the result, so it is wrong to return NUM itself.

The array ASSEMBLY has to be a global variable for the function and it is wrong to declare it locally.

The result "zero" (a string of four characters) is different from the numeric value 0.
For the "dry run" take care with column headings - vague words like "working" are not acceptable.
Don't put ASSEMBLY when you mean ASSEMBLY[i]. Add labels or line numbers to your code in (a) so that you can quote them in the dry run.

Several answers tried to announce the "not found" result too early by using code similar to this

```
for(i=0;i<MLEN;i++){
        if(NUM==ASSEMBLY[i])
                return(i);
        else
                return(0);
```

The question clearly states "dry run YOUR code..." so that marks cannot be awarded for a dry run claiming to find the correct answer from incorrect or missing code.

Question 4

4. Describe the benefits that high-productivity tools bring to software development. In particular, comment on error rates, requirements capture, and user satisfaction. Give full reasons to support your answers. **(30 marks)**

Answer Pointers
Answers similar to the following, with full reasons are expected for full marks.

- Error rates should be reduced because high-productivity tools use generators to convert tool notation into executable code. Errors that remain are shortfalls in system architecture or requirements.

- Requirements capture should be better because of incremental approaches possible with high productivity tools. Requirement errors are seen sooner, corrected sooner and increments add to a more complete system.

- User satisfaction should be increased, because of quicker visibility of product and more complete capture of requirements.

The marks were distributed as follows:

6 marks for general benefits, 8 marks for comment + explanation about error rates, 8 marks for comment + explanation about requirements benefits & 8 marks for comment + explanation about user satisfaction.

Examiner's Comments
Many candidates seemed to be looking for an opportunity to write down everything they knew about CASE tools. This led to some long answers receiving relatively few marks. For example there were no marks available for listing possible *disadvantages* of high-productivity tools (e.g. cost, training required). There were no marks available for just describing how they work unless a benefit was mentioned. There were no marks for *classifying* CASE tools (e.g. upper, lower, cross-level) unless benefits were mentioned. There were no marks available for listing and describing *techniques* (examples from answers include DFD, ERD, SSADM, flowcharts, OOP, Gantt chart, etc) since the question very specifically asked about *tools*. Many answers gave the impression that a compiler is a high-productivity tool and that syntax error checking is all that there is to finding errors. Maybe some modern compilers have high-productivity components, but a compiler is a necessity, it is not a high-productivity tool that you can choose whether or not to use.

It was very disappointing to find that more than half the answers ignored the big hint in the question to structure the answer around three points (errors, requirements, user satisfaction).

There was a lack of clarity in the answers. Many answers contained something like "documentation is a benefit of high-productivity tools" as if it were not possible to obtain documentation any other way. A more careful answer would say that it was the "automatic generation of (some parts of) the documentation" that was the benefit.

There seemed to be quite a bit of repetition in the answers.

Question 5

5. *a)* Write an algorithm or code for the factorial function fac(n) where
fac(n) = n*(n - 1)*(n - 2)…3*2 *1

(Either an iterative or recursive method may be used.) **(3 marks)**

b) Using the series $e = 1 + \frac{1}{\text{fac}(1)} + \frac{1}{\text{fac}(2)} + \frac{1}{\text{fac}(3)} + \ldots + \frac{1}{\text{fac}(n)}$

Develop a program to show that the difference between the convergent value for *e* =2.7182818 and the value accumulated for (n) terms of the above series is less than 1 /[fac(n - 1)*(n - 1)].

(9 marks)

Answer Pointers

a)

```
{recursive factorial function}
FUNCTION fac(N:INTEGER):INTEGER;
BEGIN
   IF N>0 THEN     fac:=N*fac(N-1)
   ELSE
     IF N=0 THEN    fac:=1
     ELSE
     BEGIN
        WRITELN('USE A POSITIVE INTEGER:negative factorials not defined');
        fac:=-MAXINT
     END
```

b)

```
PROGRAM eval_e(INPUT,OUTPUT);
CONST e = 2.7182818;
VAR fval,n,ct :INTEGER;  sum,term,recip,diff:REAL;

{recursive or iterative  factorial function belongs here}

BEGIN
   WRITELN('evaluate e using factorials');
   WRITELN('input the number of terms (n)');   READ(n);
sum := 1.0;
FOR ct := 1 TO n DO
          BEGIN
          fval := fac(ct);
          recip := 1/fval;
          sum := sum + recip
          END;
term :=fac(n - 1)*(n - 1);
recip := 1/term;
diff := ABS(sum - e);
WRITELN('recip. sum ',sum:10:8,' e-value ',e:10:8);
WRITELN('difference ',diff,'    recip. term ',recip);
IF recip > diff THEN WRITELN('1/fac(n-1)*(n-1) is greater')
                  ELSE WRITELN('series difference is greater')
END.
```

Examiner's Comments

This was moderately popular as a last question, when candidates wrote out memorized code for factorial(n), worth only three marks. Few attempted any algorithmic development before launching into code for part (b). Again, just a few got it completely right.

Question 6

6. Hill gradient signs at the roadside show the approaching slope either as a percentage (P %) or as a slope of 1 in G, where G is an integer. Thus a slope of 20% could also be shown as 1 in 5.

a) Derive the general relationship between the percentage P and the nearest integer slope G. **(3 marks)**

b) Write an interactive program which prints a table of values of P (in increments of 1) between input start and finish values, and the corresponding values of G, rounded to the nearest integer value. **(9 marks)**

Answer Pointers

a)

A rise of P percent means a rise of P on a base of 100

Hence a rise of 1 unit in 100/P .

This can be expressed as 1:: 100/P where P is rounded to the nearest integer.

So generally G = ROUND(100 / P)

b)

```
100 PRINT "hill slope/gradient program"
110 PRINT "Dr J. Wilford March 2004"
120 PRINT "INPUT starting value for % incline"
130 INPUT start
140 PRINT "INPUT end value for % incline"
150 INPUT fin
160 PRINT " table of slopes and corresponding gradients"
170 PRINT "percent"; TAB(17); "slope"
200 FOR P = start TO fin STEP 5
205 IF P = 0 THEN
PRINT " 0"; TAB(17); " 0"
ELSEIF P <> 0 THEN
LET G = CINT(100 / P)
PRINT P; "    1  :"; TAB(17); G
END IF
230 NEXT P
240 END
```

Examiner's Comments

Part a) was done poorly as few bothered to 'derive' the result by simple proportion. Very few realized that 'round' function is necessary to express G as an integer, as stated in the question. Likewise in part b) (where rounding was again emphasized) most did not write an algorithm before launching straight into code.

Question 7

7. A school keeps daily meteorological records on a serial file. Each record has the following data items:

Temperature/deg. C : 2-digit integer
aspect : (fine or wet)
rainfall/mm : real number
wind direction : one of (north, south, east, west)
wind force/Beaufort scale : 2-digit integer 0<= wind<=12
observer's initials : (between 1 and 5 characters)

a) Write an appropriate description for such a record in a suitable programming language. State which language you are using. **(4 marks)**

b) Write a program fragment which opens such a file named '*August05*' and which counts and prints how many days were wet and windy, i.e. had aspect = 'wet' and wind force greater than 4. **(8 marks)**

Answer Pointers

a)

```
Oneday : RECORD
                Temperature : INTEGER;
                Aspect : (wet, fine);
                Rainfall : REAL;
                Wind_direction : (north, south, east, west);
                Wind_force : INTEGER;
                Observer: PACKED ARRAY [1..5] OF CHAR
           END;
```

b)

```
VAR  onemonth : FILE OF oneday;    aday : oneday :

                  RESET onemonth;
                  recordct = 0
                  WHILE NOT EOF(onemonth) DO
                           BEGIN
                           READ (onemonth, day);
                           WITH aday DO
                           IF (aspect = wet) AND (wind_force > 4) THEN recordct := recordct + 1;
                           END;
                  WRITELN(recordct :2, 'wet and windy days');
                  CLOSE onemonth
END;
```

Examiner's Comments

This was generally done well. Those who failed to declare a record (or structure) were penalized as the question stated that one was required. A few candidates wasted time by writing an elaborate interactive input sequence, not realizing that the records are read from the file <*August05*>.

Question 8

8. *a)* Write a linked list data structure capable of holding one data item and one pointer. How is this shown diagrammatically? **(3 marks)**

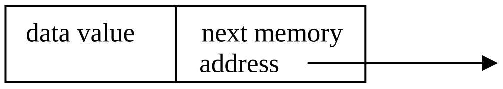
b) Write pseudocode to set up such a linked list using a '*while'* or '*repeat'* loop. Alongside each instruction show diagrammatically the pointer movement so made or change in memory effected.

State which programming language you are using. **(9 marks)**

Answer Pointers

a)

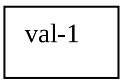
```
TYPE    ptr = ^node;
        node = RECORD
                  data : (any single-valued type);
                  next : ptr
                END;
```



b)

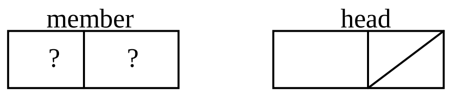
head $\leftarrow$ NIL

Read(item)

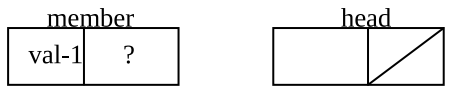


```
WHILE  item not equal to terminator DO        [enter loop]
```

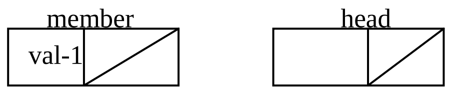
```
NEW (member)
```



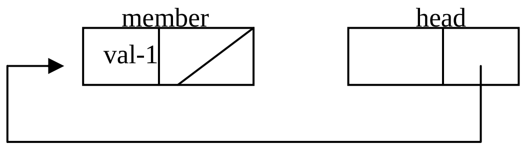
```
member^.data ← item
```



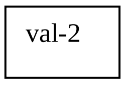
```
member^.next ← head
```



```
head   ←member
```

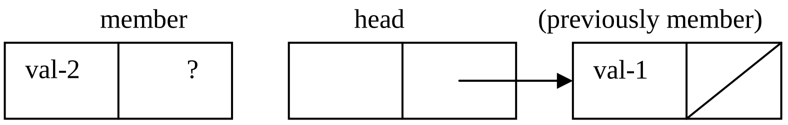


```
Read(item)
```

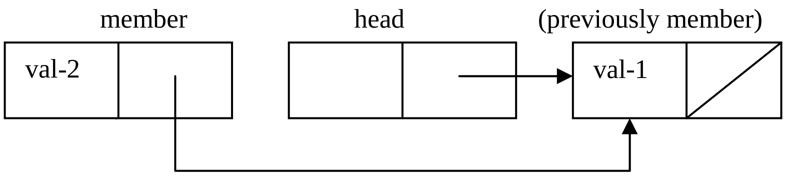


```
WHILE (item <> terminator) DO     [Loop repeats while item does not equal the terminator]
```

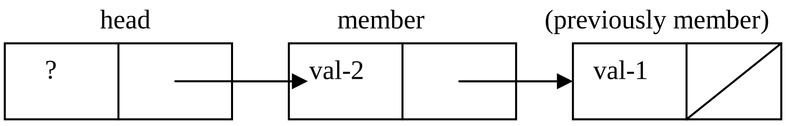
```
NEW(member)
member^data ← item
```



```
member^next ← head
```



```
head ← member
```



Examiner's Comments

This was quite popular, as is programming bookwork generally as opposed to tackling problems. The question was designed to see if candidates understood the meaning of the familiar box diagrams and what they implied for writing code. Results showed unequivocally that this is NOT understood. Nearly all the marks would have been significantly lower if the marking scheme, tying instruction to diagram had been rigorously applied. Unfortunately students see this work as an opportunity for writing out memorized code as there are only a limited number of possibilities allowed by the syllabus. Good marks were given for correct set-up code and the associated diagram for building the linked list.

Tutors should ensure that their students fully understand what is implied by the box diagrams for associated code. If they do so, this area of programming is not harder than any other. It is not intended to afford opportunity for feats of memory, which are undone if questions use different terms from those remembered. Such memorized code is rarely useful in the workplace as well.

Question 9

9. *a)* Describe TWO different ways to use a sequential file. Give reasons for your answer. (**8 marks**)

b) Make a diagram that shows the detailed sequence of operations for ONE of the ways you describe in part *a)*. **(4 marks)**

Answer Pointers

This question was interpreted in a large number of ways. Some answers chose to write about two file organisations (e.g. sequential, indexed sequential), some about two application areas (e.g. master file, backup file) some about two individual applications (e.g. producing monthly invoices, or some other batch processing task).

a) All the kinds of answers mentioned above were awarded marks if the way of using the file was named and explained

b) The simplest diagram capable of scoring full marks would be a diagram of a sequence of records representing the file, with a pointer showing the reading position, re-drawn (say) three times to show the pointer advancing record by record.

The marks were distributed as follows:

2 marks for use and 2 marks for explanation, another 4 marks for another use and explanation.

To obtain 4 marks in part b) there had to be a diagram, it had to be clearly linked to one of the uses mentioned in (a), be clearly labelled, and show a <u>sequence</u> of operations.

Examiner's Comments

(a) There was no excuse for only describing ONE use, which of course meant only half marks.

It seemed that there was considerable confusion between a serial file (natural arrival order) and a sequential file (ordered by key)

There was much confusion in the written answers between the words file/record/key. Which word is correct in the sentence "The [file] / [record] / [key] was looked up in the index"?

(b) Many answers showed a picture of a sequential file, but not a detailed sequence of operations.

The diagram needed to be linked with one of the uses in (a) so there were no marks for drawing a second diagram linked to the other use.

Question 10

10. Design the GUI for a web system to be used by an e-commerce operation that sells electrical goods. Include in your design THREE different types of interface element which make the process of e-commerce easier and more attractive. Explain your design fully, using diagrams. **(12 marks)**

Answer Pointers
Expect a window design using interface elements like frames, buttons, menus (drop-down lists), check boxes, radio buttons, text boxes, etc. The design should be relevant to electrical goods. Three interface elements should be singled out and named. For each of the three elements it should be stated what interface interaction that they are best used for in general and what they are used for in this design.

Marking: 3 marks for overall window design, 3 marks for each interface element (name, drawing, general or specific use). Deductions for inappropriate use of interface elements. Deductions for acceptable design with no explanation.

Examiner's Comments
The wording of the question tried to emphasize that it was the GUI that was important. Unfortunately there were many answers which described features of the overall website that were not related to the interface. For example describing the importance of the customer satisfaction policy or guarantee did not gain any marks.

Possibly the most commonly selected three interface elements were radio/option buttons, drop-down list/menu, check boxes. Other interface elements were accepted; however there were many mistakes or inaccuracies in answers describing frames. Candidates need to take care in describing how many frames/panes make up a *single* window. Equally candidates should be careful in describing the typical frame interaction of a click in one frame causing the content of a *different* frame to refresh.

Question 11

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

a) A Spreadsheet. **(4 marks)**
b) A Project planner. **(4 marks)**
c) A Disk File Index. **(4 marks)**

Answer Pointers
a) A Spreadsheet is application and models numerical and financial situations, specialising in presentation and 'what-if' modelling.
b) A Project planner is application, and sequences tasks for execution in a project, specialising in modelling constraints and critical paths.
c) A Disk File Index is system, and helps to locate disk records faster.

Marking: for each part, 2 marks for description and 2 marks for system/application

Examiner's Comments
Several answers missed capturing the essence of the software concerned. It is not enough to state that a spreadsheet allows you to enter numbers into rows and columns. It is necessary to mention performing calculations. Likewise it is not helpful to focus on secondary facilities and to give the impression that you think that the principal task of a spreadsheet is to provide database facilities or word processing facilities.

It helps to give an example application if known (e.g. M/S Excel, M/S Project)

It was a bad mistake, found in too many answers, to omit the statement of whether the software was "application" or "system". This reduced the available marks from 12 to 6.

To say that a project planner is used for project planning does not count as describing the function of the software.

Question 12

12. Briefly describe the type of testing you would use for:

a) module tests **(6 marks)**

b) validation testing **(6 marks)**

Give reasons for your answers.

Answer Pointers

a) For module/unit tests, use logic tests (a.k.a. white box) first, then check outputs are correctly produced.

b) For validation testing, test only that interfaces generate and accept correct data, and produce fail messages for incorrect data (a.k.a. black box)

Marking: (a) 3 marks for the name/type of testing and 3 marks for the reason marks available. (b) 3 marks for the name/type of testing and 3 marks for the reason.

Examiner’s Comments

Try not to treat the question as an excuse for writing everything you know about testing. For example the question does not ask which type of testing is fastest and so this does not form part of the answer.

Many answers read like an encyclopaedia of testing methods - there was white box test, glass box test, transparent box test, grey box test, black box test, top-town test, bottom-up test, sandwich test, system test, recovery test, security test, acceptance test, regression test, performance test, stress test, statistical test, virus test, integration test, alpha test, beta test, ...

The question asks you to choose a type of testing, so explaining that you need to find someone who is familiar with the programming language used - or the actual code - or both is not relevant. It does not ask what the test achieves or what testing there is still left to do.

When discussing white box testing it is necessary to speak about the *source* code being available. It is not enough to say that the code needs to be available because this could be high-level or low-level code.

When discussing black box testing many candidates were able to say that the actual output for some data was to be compared with the expected output - but hardly anyone was able to say how the tester found out what the "expected output" should be.

Some candidates mistook this for a question about validation checks on data as performed on data entered into text boxes on a web form.

Many candidates had obviously learned the difference between "building the right software" and "building the software right" and then spoilt their answers by using the word *write* instead of *right*.