EXAMINER'S COMMENTS
Certificate: Software Development, April 2007

General
A significant number of candidates have no grasp of this area of work and gained marks lower than 20%. In particular, algorithmic development is particularly weak.
Flowcharts should NOT be used for program development as they have been superceded by better methods.
It is a waste of time merely to copy out code as given in a question.
Illegible question numbers are a nuisance to the examiner. Likewise parts of a question scattered over the answer book with no indication as to where they are.
Crossing-out of whole pages of work also wastes time. Students must learn to answer questions without needing extensive rough work.
Nearly all programming answers were prefaced by 'uses crt' or '#include <iostream.h>' These are not required in a problem answer in an examination and suggests that students do not know what these directives do.

SECTION A

Question 1

The annual flood of the river Nile is important to the prosperity of countries such as Sudan and Egypt. Readings are taken of the river level at fixed stations along the Nile bank every hour and collected in a serial file. Each record has the following fields:

Station name (30 characters)
Station Reference (8-digit integer)
Date (ddmmyy) where dd = day digits, mm = month digits, yy = year digits
Followed by 24 pairs of values, each having hour of reading (24-hour clock) reading of the river level (real number)

a) Write a suitable description of this record in a language of your choice. State which language you are using. **(5 marks)**

Answer Pointers – 1a

In PASCAL a suitable record description is as follows:

```
TYPE    level_readings=RECORD
                                time:INTEGER;
                                level:REAL
                           END;
        River_Record = RECORD
                                station_name:String[30];
                                station_ref:INTEGER;
                                date:INTEGER;
                                level_data:ARRAY[0..23] OF level_readings
                           END;
```

Examiner's Guidance Notes – 1a

Generally done well, but many candidates only attempted this part, restricting their mark to a maximum of 5. Few realized an array is needed to hold the 24 river level readings. It is important that those who don't use Pascal or C **do** state what language they are using.

b) Write ONE typical record which could be used as test data. **(3 marks)**

Answer Points – 1b

station_name JUBA
station_ ref 00:12\42:40 {latitude/longitude suitable ref}
date 220506 {all-digit date convenient for processing}

time/hours	0:0	1:0	2:0	▨	21:00	22:00	23:00
level/metre	5.260	5.263	5.339	▨	5.789	5.817	5.852

Examiner's Guidance Notes – 1b

Any plausible values were accepted. Candidates were not expected to think up 24 separate pairs of time/level readings but merely to indicate that 24 such pairs were expected. This part was not understood by many candidates who gave an input sequence or repeated the declaration of **a).** Some students had no idea of what likely values would be expected here as test data.

c) The following initial algorithm is for a program intended to detect when the flood is beginning:

```
FOR each station from first to last DO
      READ one record from the file
      FOR each hourly reading in the record DO
            Obtain difference between readings at current hour and the previous hour IF the
            water has risen THEN
                  PRINT the increase in river height
            IF (three successive differences are positive)
                  THEN Calculate the rate of river rise as an hourly percentage
      ENDFOR
ENDFOR
```

Develop this algorithm to the point where coding would be straightforward. Use the record structure you have written for part (a) but do not copy it out again. State your intended target language. **(22 marks)**

Answer Pointers – 1c

Algorithm development

```
OPEN  niledata file for READING
SET test as suitable level difference value
FOR pos := 0  TO  timend   DO
      READ (niledata, onerec)
      Oneday[pos]. time := onerec.time
      Oneday [pos]. level := onerec.level
ENDFOR
FOR pos1 := 1 TO timend DO
```

```
(*Obtain difference between readings at current hour and the previous
hour *)
      level_diff[pos] := Oneday[pos].level – Oneday[pos-1].level
ENDFOR
IF pos > 2 THEN
         IF (level_diff[pos] > test) AND (level_diff[pos-1] > test)AND
            (level_diff[pos-2] > test)    THEN
                  PRINT “possible Nile flood start”
      FOR pos1 := pos-2 TO pos PRINT level differences
ENDIF
ENDPROG
```

<u>Further development in Pascal</u>

```
ASSIGN(Niledata,'C:\PROPAS\SOURCES\Niledata');
RESET(Niledata);
FOR pos := 0 TO 23 DO
  BEGIN
  READ(Niledata,level_rec);
  WRITE('read ',pos,' Record...');
  One_day[pos].time := level_rec.time;
  One_day[pos].level := level_rec.level;
  WRITELN(One_day[pos].time,One_day[pos].level);
  END;
  (* get level differences *)
FOR pos := 1 TO 22 DO
      BEGIN
      level_diff[pos] := One_day[pos].level  -  One_day[pos - 1].level;
      WRITELN(pos,level_diff[pos]);
            IF pos>=2 THEN
      IF (level_diff[pos] >= test AND (level_diff[pos-1] >= test) AND
                  (level_diff[pos-2] >= test) THEN
            BEGIN
            WRITELN('possible Nile flood start');
            WRITE('      time ');
            FOR pos1 := pos-2 TO pos DO WRITE(One_day[pos1].time);
      WRITELN;
            WRITE('     level ');
            FOR pos1 := pos-2 TO pos DO WRITE(One_day[pos1].level);
      WRITELN;
            WRITE('difference ');
            FOR pos1 := pos-2 TO pos DO WRITE(level_diff[pos1]);
      WRITELN;
            END
      END;
CLOSE(Niledata,FALSE)
END.
```

Examiner’s Guidance Notes – 1c

Very poorly done. ‘Added value’ was wanted; no marks were gained for writing out the given algorithm. Some gave lengthy interactive input sequences when the question states that the data is on a serial file. Many copied out the record declaration here despite the question instructing them not to do so. Others wasted time on lengthy variable declarations which do not aid algorithm development. Indeed, this should not be done until the coding stage as some languages (like BASIC) do not need them. It was also unwise to leave this question until last as candidates invariably ran out of time.

Question 2

a) Dry run the following pseudocode with input values x = 18, y = 36.
All the variables hold integer values. **(15 marks)**

```
Line number       Code
      1           FUNCTION ged(u,v)
      2           REPEAT
      3           IF u < v THEN
      4                 t := u
      5                 u :=  v
      6                 v := t
      7           ENDIF
      8           u := u - v
      9           UNTIL u = 0
      10          ged := v
      11          ENDFUNCT

      12          BEGIN {top level}
      13          PRINT  ‘EXECUTION STARTS HERE’
      14          PRINT 'Input top and bottom values of fraction'
      15          INPUT x , y
      16          z := ged(x , y)
      17          PRINT 'Greatest common factor = ',z
      18          x := x DIV z
      19          y := y DIV z
      20          PRINT 'reduced fraction ',x,' / ',y
      21          ENDPROG
```

Note - DIV is integer division; thus 5 DIV 3 = 1.

Answer Pointers

The tabular method (shown below) is the best method but it is not compulsory. Marks were awarded (up to 4) for correct choice of variables and conditional expressions for columns. In particular, u, v, t and loop conditions were essential. Mistakes in the table were only penalized once despite this resulting in further incorrect values.

line	Code	x	y	z	u	v	t	IF	UNTIL	Output
12,13	PRINT	?	?	?	?	?	?			
14	PRINT									Input top,bottom
15	INPUT	18	36							
16	Funct call									
1	Transfer				18	36				
2	REPEAT									
3	IF							True		
4	Assign						18			
5	Assign				36					
6	Assign				36	18				
7	ENDIF									
8	Assign				18					
9	UNTIL								False	
2	REPEAT									
3	IF							False		

4	ENDIF									
5	Assign				0					
9	UNTIL								True	
10	Assign									Ged=18
11	ENDFUNCT									
16	Assign			18						
17	PRINT									Gr com fact = 18
18	DIV	1								
19	DIV		2							
20	PRINT									Reduced fract = ½
21	END									

Examiner's Guidance Notes

A popular question, the full range of marks was used.
a) Nearly all students can do the dry-run part using the tabular method. Fewer candidates started execution on line 1 this time: the instruction PRINT 'EXECUTION STARTS HERE' was meant to help them avoid this. Weaker students made mistakes like assigning values to variables (u, v, t) on line 1 when this is not done until later on in the code.

b) Write brief notes on the use of identifiers and comments in the code. **(4 marks)**
The variables have poor names which convey no meaning about their purpose. In particular, the identifiers (such as u, v, t, x, y) had single letter names which convey nothing about what they hold; more meaningful names should be used.
There are no comments in the code, especially in the function. This particularly needs the comment "this code works out the greatest common denominator of two integers".

c) Re-write the code so that values of u are printed on entry an exit to the function. Incorporate any improvements which would make the code more readily understood by programmers.
(11 marks)

```
PROGRAM gcdtest(INPUT,OUTPUT);
{finds greatest common denominator for fraction top / bottom}
VAR top,bottom,result:INTEGER;
FUNCTION gcd(a,b:INTEGER):INTEGER;
{returns the greatest common divisor of fraction a/b}
BEGIN
 {prints current function param.}
WRITELN('top ' , a:3,'  bottom ', b:3);
IF b = 0 THEN gcd := a
          ELSE gcd := gcd(b,a MOD b)
END;
BEGIN {top level - execution begins here}
WRITELN(' factor fraction program');
WRITELN(' enter numerator/top of fraction');READ(top);
WRITELN('enter denominator/bottom of fraction'); READ(bottom);
result := gcd(top,bottom);
WRITELN('factor = ',result:3);
top := top DIV result;   bottom := bottom DIV result;
WRITELN('simplified fraction  ',top :2,' / ',bottom:2)
{top, bottom, result are more meaningful names}
END.
```

Marks were given for ‘added value’ imparted to the code by better identifier names and incorporating several comments. No marks were given to those who merely copied out the supplied code unchanged. Others forgot to print the value of u inside the function.

Question 3

a) What are the operations usually applied to the data structure called a stack? **(5 marks)**

b) Write an implementation of a stack (data structure and operations) in a programming language that you know. The values to be stored in the stack will be integers. You may assume that there will never need to be more than 20 values stored in the stack at any one time. **(15 marks)**

c) Using your implementation, create a program which reads 10 integers from the input and outputs them in reverse order. **(10 marks)**

Answer Pointers

a)
push(e) - push a new element e onto the stack
pop() - pop the top element off the stack and return as result
peep() - return the element on the top of the stack (leaving the stack unchanged)

b)
[Answers in any programming language accepted - even pseudocode - the answer given below is in C]

```
int stack[20];
int top=-1;
void push(int e){ top=top+1; stack[top]=e;}
int pop(){ int e; e=stack[top] ); top=top-1; return(e); }
int peep(){ return(stack[top]); }
```

(c)

```
int stack[20];
int top=-1;
int e,i;
void main(){
        for(i=1; i<=10; i++){ scanf("%d", &e); push(e); }
        for(i=1; i<=10; i++){ e=pop(); printf("%d", e); }
}
```

Examiner’s Guidance Notes

a) Answers should really include push & pop which were worth two marks each. The fifth mark was awarded in addition for any one out of peep(), initialize(), isempty(), isfull().

Here and also in (b) care should be taken that push and pop are described as opposites - pop exactly undoes the work performed by push.

Many candidates wrote about last-in-first-out (LIFO) and first-in-last-out (FILO) which was awarded one mark in the absence of any of the operations above - but note that FIFO and LILO refer to a queue and were thus not acceptable.

b) There were three marks for setting up the data structure and four marks for each of the three subroutines/functions named in (a).

Nearly all candidates who attempted this part chose to implement using an array (as anticipated). It was not enough just to write some (main program) code to place values in an array - it had to be achieved with subroutines named as in (a).

Many answers made mistakes with pop(). One mistake was to concentrate on deleting the element from the stack and omit to return the top element as a result. Another mistake was to get the sequence of decrementing of the stack top pointer and obtaining the stack top value the wrong way round, so that pop() would not exactly undo the work performed by push().

c) Two loops were needed, the first one has to execute 10 times and each time read a value and push it onto the stack. The second one also executes 10 times and each time pops a value off the stack and prints it.

Code that 'worked' (i.e. read 10 numbers and output them in reverse order) could not gain full marks unless it operated by using the push, pop subroutines/functions introduced in part (b).

Question 4

a) A function can be recursive or not recursive. If you were reading a function definition, how would you discover if it was recursive? **(4 marks)**

b) Any function is allowed to have local variables, but what special arrangement has to me made for local variables of a recursive function? **(4 marks)**

c) Write down the output you would expect from the function *rec* below following the call *rec(5)*

Note: The function rec is defined in two languages (version 1 & version 2). You may use either definition to obtain your answer. **(12 marks)**

d) Suppose the programmer missed out the 'if' test and that the middle line was just written rec(p-1). What would happen now when the function was called? **(5 marks)**

e) Based on your answer to (d), give one important rule for a programmer writing a recursive function. **(5 marks)**

Version 1	Version 2

void function rec(int p){
   printf("starting %d \n",p);
   if(p>0) rec(p-1);
   printf("finishing %d \n",p);
}
	PROCEDURE rec(p as INTEGER)
	PRINT "starting " p
	IF p GREATER THAN 0 THEN rec(p – 1)
	PRINT "finishing" p
	ENDFUNCT

Answer Pointers

a) A function is recursive if it has a call of itself within its own body

b) When recursive, a 'new' version of the function may need to start execution before the existing version has finished so a separate copy of the local variables is needed for the new call

c) The output is:

```
starting 5
starting 4
starting 3
starting 2
starting 1
starting 0
finishing 0
finishing 1
finishing 2
finishing 3
finishing 4
finishing 5
```

d) The function is now infinitely recursive and will run indefinitely or until stack overflow or some similar hardware restriction is encountered.

e) There must always be at least one non-recursive route through the body of a non-recursive function.

Examiner's Guidance Notes

a) Many students were not prepared for this question as shown by incorrect answers like "A function is recursive if it has a loop in it", "A function is recursive if it has a conditional in it". A correct answer was worth four marks.

b) Hardly anyone was able to answer this part correctly. A correct answer was worth four marks.

c) The most common answer contained a countdown from 5 to 1 or 5 to 0 and went no further. However there is both a 'starting' print and a 'finishing' print statement in the function so these messages must come out in matched pairs with any output from the recursive call being sandwiched in between. There was a mark for each correct word & number combination. Correct numbers on their own were only awarded half marks. There were 12 marks available so the answer "starting 5 starting 4 starting 3 starting 2 starting 1 starting 0" would get half marks and "5 4 3 2 1 0" would get a quarter of the marks.

It was not clear why candidates wrote code as part or all of the answer to this section.

d) There were many wrong answers that showed a misunderstanding of how the code worked. Many candidates wrote wrong answers such as "With the 'if' test removed the results will be the same" or "With the 'if' test removed the function will be called just once". A correct answer was worth five marks.

e) Not surprisingly, following the wrong answers given for previous sections, this was not well answered. A correct answer was worth five marks.

SECTION B

Question 5

a) Write a program which prints the mean number of weeks (**mean)** in every month for a non-leap year. Thus January has 31 days hence its mean = 31/7 or 4.4285 weeks.
meanwk is calculated by adding the **mean**s for January to December together and dividing the total by 12.

b) Calculate the standard deviation (S) from the formula given below:

$$S = \text{SQRT}\left[\sum_{i=\text{Jan}}^{\text{Dec}} (\textbf{mean}_i - \textbf{meanwk})^2 / 12 \right]$$

where **meanwk** is calculated by adding the **mean**s for January to December together and dividing the total
by 12.

Information:
January, March, May, July, August, October, December have 31 days
April, June, September, November have 30 days
February has 28 days. **(12 marks)**

Answer Pointers

An essential feature of the program must be an association of the month name (January, February, March,…,December) with the number of days in it. The simplest method is to use an integer as shown in the table below. The table also contains how many days are in the month.

Month	January	February	March	April	…	December
Integer	1	2	3	4	…	12
Days	31	28 or 29	31	30		31

The simplest association is to use SET constants. Thus
[1,3,5,7,8,10,12] are the month numbers of those months which have 31 days
[4,6,9,11] likewise whose months have 30 days.
We must also deal with FEBRUARY with 28 days except when it is a leap year when it has 29 days. A leap year occurs when the year is exactly divisible by 4 or when
<ayear> MOD 4 = 0.
A nuisance here is that we need to keep the mean weeks for every month for use in the standard deviation calculation. The formula implies that this mean is kept in an array mean[month]
We can now write the algorithm:

```
INPUT ayear
IF ayear MOD 4 = 0 THEN leayyr := TRUE   ELSE  leapyr := FALSE
Set totals to zero
FOR mth := 1 TO 12 DO
      IF mth IN [1,3,5,7,8,10,12] THEN meanwk[mth] := 31/7
          ELSE IF mth IN [4,6,9,11] THEN meanwk[mth] := 30/7
            ELSE IF mth = 2 THEN
                    IF leapyr THEN meanwk[mth] := 29/7
                              ELSE meanwk[mth] := 28/7
      PRINT 'mean days in ',mth, ' is',meanwk
      ADD meanwk  TO tot
(* mean of means *)
avmean := tot / 12
(*Calculate standard deviation *)
```

```
Sum := 0.0
FOR mth := 1 TO 12 DO ADD SQR(meanwk[mth] - avmean) TO Sum
(* calculate mean of means *)
S := SQRT(Sum/12)
PRINT 'av.mean = ',avmean,' Standard deviation of mean = ',S
ENDPROG
```

A simple development of the algorithm uses a CASE statement to associate the number of days in the month with the mean number of weeks in it. This replaces the sets used earlier.

```
CASE mth OF
1,3,5,7,8,10,12:meanwk[mth] := 31/7
4,6,9,11:meanwk[mth] := 30/7
2: IF leapyr THEN meanwk[2] := 29/7 ELSE meanwk := 28/7
ENDCASE
```

The candidates were expected to develop such a program as their answer.

A more elegant approach uses enumerated type (Jan,Feb,Mar…Dec) to represent the months. This is more meaningful than (1,2,3…12) as used earlier.

```
TYPE monthtype = (Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec)
```

The main loop can be controlled by
`FOR a month := Jan TO Dec` instead of integers 1..12 .

The month name and the number of days in it can be associated by a function getdays() which returns the number of days thus:

```
FUNCTION getdays(onemonth:monthtype;leapyr:BOOLEAN): INTEGER
BEGIN
      CASE onemonth OF
      Jan,Mar,May,Jul,Aug,Oct,Dec : getdays := 31
      Apr,Jun,Sep,Nov : getdays := 30
      Feb : IF leapyr THEN getdays := 29 ELSE getdays := 28
      END
ENDFUNC
```

To offset these advantages a procedure must be provided to print out the month name corresponding to its enumerated type:

```
(* procedure to print any month from emonth *)
PROCEDURE printmonth(inmonth:monthtype)
BEGIN
CASE inmonth OF
Jan: WRITE('JANUARY   ')
Feb: WRITE('FEBUARY   ')
Mar: WRITE('MARCH     ')
…………………………………………………………… {this kind of abbreviation is always
acceptable}
Dec: WRITE('DECEMBER  ')
ENDCASE
ENDPROC
```

The top level of the program becomes

```
BEGIN  (* TOP LEVEL *)
PRINT 'MEAN and SD.DEV program for weeks in a year'
PRINT 'Input the year/4 digits' INPUT ayear
leapyr := ayear MOD 4 = 0
(* find mean for whole year *)
```

```
tot := 0.0
FOR amonth := Jan TO Dec DO
      BEGIN
      ndays := getdays(amonth,leapyr)
      mean := ndays/7
      PRINT 'mean days in ' printmonth(amonth)
      PRINT ' are ',mean
      ADD mean to tot
      END;
avmean := tot / 12
(*Calculate standard deviation *)
Sum := 0.0
FOR amonth := Jan TO Dec DO
      BEGIN
      mean := getdays(amonth,leapyr) / 7
      Sum  := Sum + SQR(mean - avmean)
      END
S := SQRT(Sum/12)
PRINT 'av.mean = ',avmean,'  Standard deviation of mean = ',S
ENDPROG
```

Examiner's Guidance Notes

Various ways were tried by candidates to associate the number of days in a month with either its name or corresponding month in year, some quite ingenious, other clumsy or wrong. Hence the longer description given above. Others tried to work out the numerical answer to (b), awkward in an exam where calculators were not allowed!

Question B6

A factory makes ball bearings having 1.0 cm diameter. As part of the quality control process a sample of these is made regularly and the diameter of them measured accurately. Those whose diameters differ from 1.000 cm by up to 0.005 cm are classed as grade 1; those that differ by up to 0.05 are classed as grade 2. Others are rejected and returned for re-processing.
Write an interactive program which accepts ball bearing diameters until a negative value is input. It counts and prints how many of them are grade 1, grade 2 or rejects with corresponding percentages. **(12 marks)**

Answer Pointers

```
CONST  size = 1; {Ideal Bearing Diameter}
vgradei = 0.005;   {Variance allowed for grade 1}
v grade ii = 0.05;  {Variance allowed for grade 2} tol = IE-4; {Tolerance }

VAR
tgrade1, tgrade2, total: INTEGER;
diameter, pgradei, pgradeii, preject, variance: REAL;
BEGIN
{Initialise Counters }
tgrade1 := 0;  tgradeii := 0; total := 0;
```

```
WRITELN('input diameters of bearings or a negative value to end input :-');
READ(diameter); {Input and grade diameters}
IF diameter >= 0 THEN BEGIN
REPEAT
        total := total + 1 ;
        variance := ABS ( size -diameter );
{Test for grade 1 }
IF variance < ( vgrade1 + tol ) THEN  tgrade1 := tgrade1 + 1 ELSE
IF variance < ( v grade2 + tol ) THEN tgrade2:=tgrade2+ 1 ;
READ( diameter)
                UNTIL diameter < 0 ;
{ Calculate grade percentages }
pgradei := tgrade1 / total * l00 ;
pgradeii := tgradeii / total * 100;
preject := 100.0 - pgradei - pgradeii;
WRITELN;
WRITELN('this sample of ball bearings consists of :-');
 WRITELN('GRADE 1 = ',pgradei:7:3);
WRITELN('GRADE 2 = ',pgradeii:7:3);
{Output percentages}
WRITELN('REJECTS = ',preject:7:3)
END
END.
```

Examiner's Guidance Notes

A popular question. A surprisingly large number of answers had no condition-controlled loop; their code processed one ball bearing diameter only. Others forgot to include the percentage calculations. Generally done quite well.

Question 7

a) What does the term Random Access mean? **(4 marks)**

b) Give an example of a computer device which is Random Access. **(4 marks)**

c) Give brief details of situation in which a Random Access file would ideally be used. **(4 marks)**

Answer Pointers

a) Random Access means that any available address can be reached equally quickly

b) Main Memory, Disc Store [at least in comparison to tape-based device]

c) A typical scenario is where an individual, fast, online response is required [as opposed to batch processing] - as provided by modern database systems - so any online query system [e.g. what's my balance?]

Examiner's Guidance Notes

a) Many answers were unsatisfactory because they used the same words in the answer as in the question (e.g. "Random access means accessing randomly"). When a question asks "What does it mean?" this is an invitation to the candidate to find their own words to explain a concept.

In this part of the question (where just the term Random Access is being defined) there should be no mention of actual devices or algorithms.

b) The question asks for a device so answers should mention a piece of hardware (not software). Acronyms like RAM should be written out in full.

Several answers gave the CPU as an example which was not acceptable.

c) The best answers described a fairly specific example related to real life - e.g. looking up a customer's details during a telephone enquiry.

Note that the question specifically requests a situation where one (single) random access FILE would be used. This is a situation where there is a need to randomly access any data within a single file. There was confusion in answers between this and randomly choosing to access one particular file from a filestore.

Some candidates gave 'playing music' as the example (probably having the 'random play' feature of a music player in mind) but they would be very upset to hear the individual notes of their favourite song played in a random order.

Question 8

The function Fn(x) = LN(1.0 + SIN(x)) may be evaluated using the Maclaurin series:
Fn(x) = $x - ½ x^2 + 1/6\ x^3 - 1/12\ x^4 + 1/20\ x^5 - \dots$
A finite number of terms (N) are used in an evaluation of Fn(x).

a) Write down the next two terms in this series. **(2 marks)**
The denominators are formed by 1/2, 1/3*2, 1/4*3, 1/5*4… so the next two denominators are 1/6*5 = 1/30 and 1/7*6 = 1/42 .
The sign of the term also alternates +, -, +, -… so the even powers of x have a negative sign
The next two terms are – 1/30 x^6 + 1/42x^7 .

b) Show how to calculate each term in the series. **(4 marks)**

Fn(x) =	x	- ½ x^2	+ 1/6 x^3	- 1/12 x^4	- 1/20 x^5 +	1/30 x^6	+ 1/42 x^7 …
Term (n)	1	2	3	4	5	6	7
denominator =		2*1	3*2	4*3	5*4	6*5	7*6

Thus for term (n) denominator = 1/((n(n – 1))
We can obtain each new term in the series from multiplying by sign*mult*xterm where

```
        sign := sign *(-1);
        mult := 1/((ct)*(ct - 1));
        xterm := xterm *x;
```
This is added into the evolving series by
```
        series := series +sign*mult*xterm
```

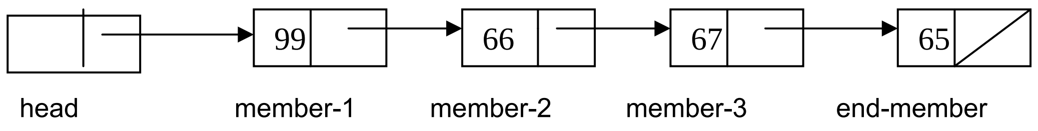
c) Write a program which evaluates Fn(x), which asks how many terms are to be used from the series. It prints this value and compares it with the value obtained using system functions for LN(x) and SIN(x). **(6 marks)**

```
(c)PASCAL PROGRAM mclurin2;
(* Evaluates ln[1 + SIN(x)] using Maclaurin Series *)
VAR sign,ct,nterms:INTEGER; x,xterm,mult,series,val,diff,fn:REAL;
BEGIN
WRITELN('Maclaurin series for ln(1 + sin(x)');
WRITELN('input how many terms'); READ(nterms);
WRITELN('input the value for <x>'); READ(x);
sign := -1;
xterm := x;
series := x;
FOR ct := 2 TO nterms DO
      BEGIN
      sign := sign *(-1);
      mult := 1/((ct)*(ct - 1));
      xterm := xterm *x;
      series := series +sign*mult*xterm
      END;
fn := 1.0 + SIN(x);
WRITELN('function evaluates LN[',fn:5:3,']');
WRITELN('function after ',nterms:3,' terms = ',series:8:6);
val := LN(fn);
WRITELN('actual value = ',val:8:6);
diff := ABS(val - series);
WRITELN('difference = ',diff:8:6)
END.
```

Examiner's Guidance Notes

An unpopular question. Most answers had the correct following members of the series, but were not able to follow this by the relation between the terms or write any code to evaluate the series.

Question 9



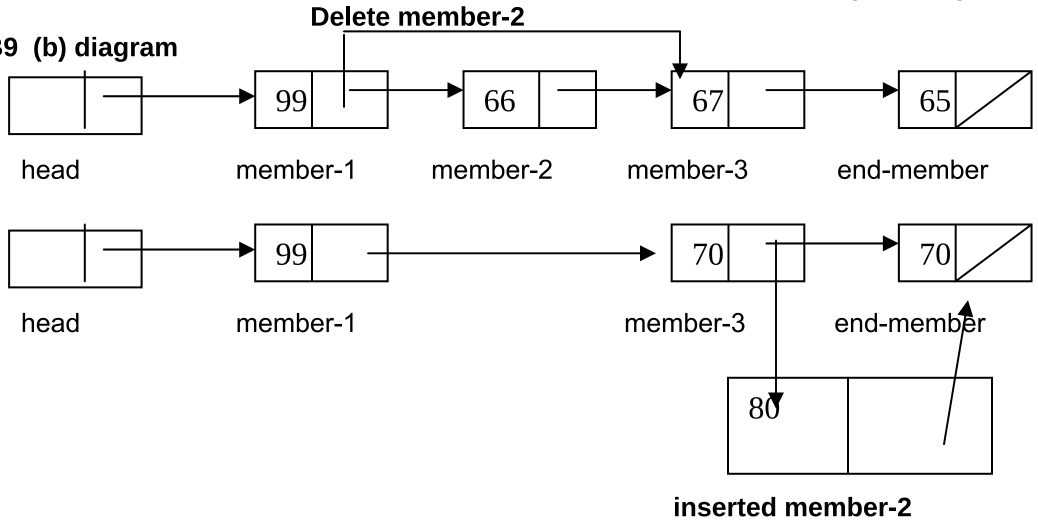
The diagram represents a linked list with a single pointer field.

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

```
TYPE   ptr = ↑node;
        Node = RECORD
        Data : INTEGER;
        Link : ptr
          END
```

b) Copy the diagram and show the necessary pointer movements to delete member-2 and insert it after member-3. **(3 marks)**

B9 (b) diagram



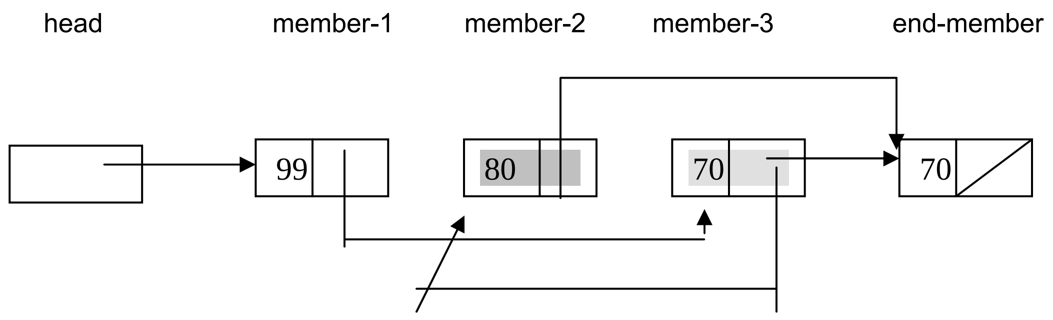
c) Write code for these pointer movements in the same language you used in part **(a).** Include the necessary variable declarations in your answer. **(6 marks)**

Answer Pointers

VARIABLES head, temp, member-1, member-2, member-3 : ptr; **(2 marks)**
<u>Deletion</u>
temp := member-2;
member-1↑.link := member-2↑.link;
DISPOSE(member-2); **(2 marks)**

<u>Insertion</u>
NEW(member-2);
member-2 := temp;
member-2↑.link := end-member;
member-3↑.link := member-2; **(2 marks)**

The questions was also interpreted as “swap member-2 with member-3” with the pointer movements shown below:



temp := member-1.link
member-1.link := member-2.link
member-2.link := member-3.link
member-3.link := temp

Examiner's Guidance Notes

Quite popular. A wide diversity of languages was used to code the linked list. Although not intended, a swap of member-2 and member-3 was allowed. Most answers had the declaration and 'swap' pointer movements but not the code derived from them. Some correct answers also had code for setting up the list, which gained no extra marks. Weaker answers had memorized code for list traversal or deletion.

Question 10

Consider the design and implementation of the search facility in a text-editor. During the process a specification will be drawn up, sometime later an algorithm will be produced and later still some documentation. With reference to this example, write brief notes on what you understand by:

a) a specification	(4 marks)
b) an algorithm	(4 marks)
c) documentation	(4 marks)

Answer Pointers

a) a specification is a concise, complete description of what is required, preferably without hints at implementation
[e.g. user should be able to specify a pattern to search for in the text, starting at the current cursor position]

b) an algorithm is a programming description of how the specification is to be achieved.

[e.g. If pattern is P and cursor is at initial position IC, set C to IC, look for P at C and if not found move C one place forward and search again. If C is beyond the end of file start C at beginning of file. If C reaches IC again, stop.]

c) documentation for the user would give information on how to use the search facility.

[e.g. which menu selection to use to start the editor, how to fill in the pattern text box, which button to press to start the search, continue the search, abandon the search, etc.]

Examiner's Guidance Notes

In each part of the question there were four marks available, three for a correct description of the topic in question and a fourth mark for relating it to the example of a text editor.

Many answers were unsatisfactory because they used the same words in the answer as in the question (e.g. "In a specification you specify...", "The documentation will be documented..."). When a question asks "What you understand by...?" this is an invitation to the candidate to find their own words to explain a concept.

Especially common in this question (but found elsewhere) was the tendency to state everything in the answer twice as if a longer answer containing duplicate material would be worth more marks. In fact this may have the opposite effect of irritating the examiner.

a) "Design" is a very bad word to use in an explanation of a specification

b) Some confusion between algorithm and final code

c) This part was answered most confidently

Question B11

a) Describe the difference between sequential and parallel programming. **(4 marks)**

b) Describe a problem where parallel programming is useful. What characteristics does the problem posses which makes parallel programming particularly suitable?

(8 marks)

Answer Pointers

a) sequential programming uses one processor and all the tasks are performed in sequence one after another

parallel programming uses multiple processors so that several tasks can be carried out simultaneously [usually with some coordination]

b) searching problems - different tasks search different parts of the search space simultaneously; numerical problems e.g. matrix multiplication - one task per element of the answer matrix

Ideally need a problem which breaks down into several (many) identical sub-problems which are independent

Examiner's Guidance Notes

A reasonably popular question but not answered in the way that was intended (see answer pointers).

a) Many answers instead spoke about the choices to be made in developing modules for a system. With a single programmer they would be developed sequentially, one module after another. With a team of programmers various modules could be developed at the same time (in parallel).

b) The question asked for an example of serial versus parallel programming, so giving a hardware-based answer along the lines of serial versus parallel connection for devices was not acceptable.

Another common misinterpretation was "running a new system in parallel with an old system that is being phased out".

Partial marks were awarded for answers which contained a clear description of the distinction between parallel and serial in some circumstance.

Candidates might like to avoid the adjective "parallely" and instead use the phrase "in parallel".

Question 12

It is possible to group software under three headings: Systems Software, Programming Software and Applications Software.

a) In which of the three groups you would put each of the following items of software

- i) Pascal Compiler
- ii) iTunes music software
- iii) Windows XP OS
- iv) Outlook express email client
- v) UNIX OS
- vi) Javascript interpreter

(6 marks)

b) Describe briefly the three categories in your own words in a way that would help someone work out in which category to place a new piece of software.

(6 marks)

Answer

a)	i) Pascal Compiler	Programming Software
	ii) iTunes music software	Application Software
	iii) Windows XP OS	System Software
	iv) Outlook express email client	Application Software
	v) UNIX OS	System Software
	vi) Javascript interpreter	Programming Software

b)
Systems software concerned with shielding the user/programmer from the details of the hardware

Programming software concerned with providing facilities for programmers to create new software
Applications software offers users pre-packaged solutions for a specific task

Examiner's Guidance Notes

Generally popular and reasonably well answered

a) one mark for getting each category right

It was puzzling to see so many answers put the Pascal Compiler and the Javascript Interpreter into different categories

b) two marks for each of three descriptions

Very few attempted to answer the question as worded - i.e. "to help someone work out...". Almost all the answers were just general descriptions of the three software headings.