Answer THREE questions, at least ONE from EACH of Sections A and B.

SECTION A

1.

a) In algorithmic analysis, what is meant by the term elementary operation ?

[6 marks]

b) Why is cost measured in terms of elementary operations rather than in CPU time?

[7 marks]

c) Briefly describe how best case analysis differs from worst case. Distinguish clearly between the properties of a problem and those of a particular algorithmic solution. If possible, provide an example of best case analysis to illustrate your argument.

[9 marks]

d) *Best case for squaring a matrix.*
A = A[1..n, 1..n] is an nxn matrix with integer elements a_{ij} (i,j = 1..n).

(i) Suppose that the multiplication of two matrix elements can be regarded as an elementary operation. Write down and justify an expression which gives a lower bound on the number of such operations which must be performed by any algorithm which *squares* the matrix A ($A \rightarrow B = AxA$).

[4 marks]

(ii) Suppose that the a_{ij} are of such a size that the multiplication of two matrix elements is itself now of significant cost. Write down and justify a expression for a lower bound on the number of *single bit* operations to be performed in this case.

[7 marks]

2.

a) Give a mathematical definition of the order notation

$$f(n) \in O(g(n))$$

and explain how this concept relates to the algorithmic idea of worst case analysis.

[9 marks]

[Question 2. cont. over page]

TURN OVER

[Question 2. cont.]

b) Are the following statements true or false?

(i) $n^2 \in O(n^3)$

(ii) $n + \frac{1}{n} \in O(n)$

(iii) $3^n \in O(2^n)$

In each case give a careful argument based on the mathematical definition of O-notation.

[12 marks]

c) Consider the following program fragments. In each case work out f(n), the exact number of unit-time operations performed, as a function of the input size n, then simplify your final answer using O-notation. Where appropriate you may use the summation formulae

$$\sum_{i=1}^{n} i = \frac{n}{2}(n+1) \qquad \sum_{i=1}^{n} i^2 = \frac{n}{6}(n+1)(2n+1)$$

(i)
```
for i := 1 to n do
    for j := 1 to i do
        for k := 1 to (i + j) do
            {an operation requiring unit time}
```

[6 marks]

(ii)
```
for i := 1 to n do
    for j := 1 to i do
        for k := 1 to (2*j – i) do
            {an operation requiring unit time}
```

[6 marks]

CONTINUED

3.

a) Solve the following recurrence relations, simplifying your final answers using O-notation (you may assume that n is a power of 2 where appropriate):

(i) $f(0) = 2$

$f(n) = 5f(n-1) - 4, \quad n > 0$

[4 marks]

(ii) $f(0) = 2$

$f(1) = 5$

$f(n) = 5f(n-1) - 4f(n-2), \quad n > 1$

[4 marks]

(iii) $f(0) = 1$

$f(1) = 2$

$f(n) = 4f(n-1) - 4f(n-2), \quad n > 1$

[4 marks]

(iv) $f(1) = 3$

$f(2) = 9$

$f(n) = 5f(\frac{n}{2}) - 4f(\frac{n}{4}), \quad n > 2$

[6 marks]

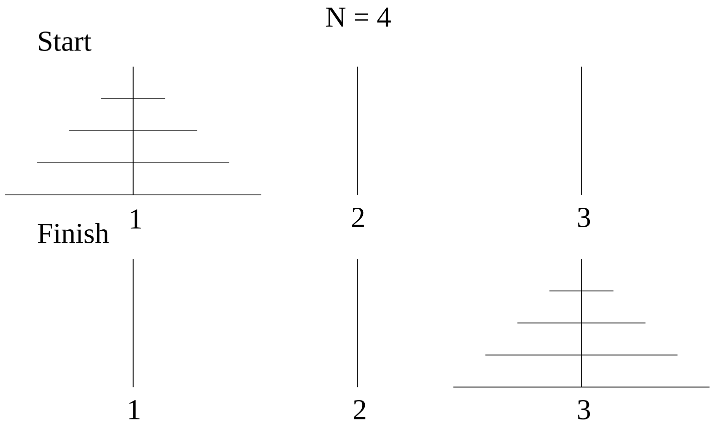
[Question 3. cont. over page]

TURN OVER

[Question 3. cont.]

b) *The Towers of Hanoi.*
The problem involves transferring N rings of graduated sizes from a starting peg (peg 1) to a final peg (peg 3), using an auxiliary peg 2 to hold the rings in transit, moving *one ring at a time* and *never allowing a larger ring to rest on top of a smaller one*.

The illustration below shows the starting and desired final ring positions for the 4-ring problem:



The problem can be solved by calling the procedure Hanoi(N, 1, 3), where Hanoi(n, i, j) moves the n smallest rings from peg i to peg j, and is defined recursively by

```
{Procedure Hanoi(n, i, j)}

if n > 0 then
    begin
    Hanoi(n-1, i, 6-i-j);
    {move a ring from i to j}
    Hanoi(n-1, 6-i-j, j);
    end;
```

(i) Write down a recursive formula for $h(n)$, the number of times a ring is moved by Hanoi(n, i, j), where $0 \le n \le N$.

[4 marks]

(ii) Solve this recurrence to show that the number of moves required to solve the N-ring Towers of Hanoi problem is $2^N - 1$.

[4 marks]

END OF SECTION A