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**?

[4 marks]

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

[4 marks]

c) Consider the following procedure to calculate the factorial of n (n!):

```
if (n == 0)
    Factorial = 1
else
    Factorial = n*Factorial(n-1)
```

(i) If comparisons and multiplications each are considered to take unit time, show that the above procedure is linear in the number of such operations performed.

[5 marks]

(ii) Explain why the result of the above analysis might be misleading. How might the analysis be improved?

[4 marks]

[Qu.1 cont. over page]

TURN OVER

[Qu.1 continued]

d) Consider the following procedure ’Mystery’, which depends on two integer variables m and n:

```
if (m == 1)
        if (n == 1)
                // do something requiring unit time
        else
                {
                // do something requiring unit time
                Mystery( n/2, 1 );
                }
else
        {
        // do something requiring unit time
        Mystery( 2*n, n/2 );
        }
```

Let the work required to evaluate Mystery(n,m) be given by M(n,m). You may assume that both n and m are powers of 2.

(i) Consider the case that m = 1. Show that

$$M(n, 1) = \log_2(n) + 1$$

[6 marks]

(ii) Now consider that $m \neq 1$. Use your result of (i) to show that in this more general case

$$M(n, m) = \log_2(nm) + \log_2(m) + 1$$

Simplify this result under ’O’ notation.

[10 marks]

CONTINUED

2.

a) Two algorithms A and B solve the same algorithmic problem. In the worst case A’s solution has a lower order of complexity than B’s, but on average the situation is reversed. Should either of these algorithms be considered ’better’ than the other? If yes, explain why; it no, describe circumstances under which each of the algorithms might be preferred.

[7 marks]

b) Two algorithms C and D solve the same algorithmic problem, C taking n days and D taking $n\log_2 n$ hours, where the parameter n describes the size of the input. How large does n need to be before algorithm C takes 5% less time than D?

[4 marks]

c) 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**.

[6 marks]

d) Are the following statements true or false? Justify your answers using a careful argument based on the mathematical definition of ’O’ notation.

(i) $(n+1)^2 \in O(n^2)$

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

(iii) $n! \in O((n+1)!)$

(iv) $\log(n^2) \in O(\log(n))$

[16 marks]

TURN OVER

3.

a) Explain the difference between **worst case** and **average case** analysis.

(i) Give an example of an algorithm whose running time on average improves on its worst case performance by a constant factor.

(ii) Give an example of an algorithm whose running time on average improves on its worst case performance by a factor which increases with the size of the input.

[6 marks]

b) Describe how the efficiency of a basic Quicksort procedure can be improved by some simple extensions to the algorithm. Give an idea of the reduction in running time (on most applications) that might be expected by adopting these modifications to the algorithm.

[6 marks]

c) Solve the following recurrence relations, simplifying your final answer using ’O’ notation:

(i) $f(0) = 0$
$f(1) = 2$
$f(n) = 4f(n-1) - 3f(n-2), \ n \geq 2$

[4 marks]

(ii) $f(0) = 0$
$f(1) = 4$
$f(n) = 4f(n-1) -4f(n-2), \ n \geq 2$

[4 marks]

d) Consider the following recurrence relations

$$\begin{aligned} f(1) &= 0 \\ f(n) &= 2\, f(n/4) + 1, \quad n \geq 4 \\[1em] g(1) &= 1 \\ g(n) &= c\, f(n/2), \quad n \geq 2, \quad c \text{ a constant } > 1 \end{aligned}$$

(i) Solve each of these recurrence relations as a function of n.

[10 marks]

(ii) For what value of the constant c would the solution for g(n) grow more slowly than the solution for f(n)?

[3 marks]

END OF SECTION A

CONTINUED