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

SECTION A

1.

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.

[6 marks]

b) Are the following statements true or false?

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

(ii) $2^{2n} \in O(2^n)$

(iii) $\log_2(2n) \in O(\log_2(n))$

(iv) $\log_a(n) \in O(\log_b(n))$, where a and b are positive integers

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

[16 marks]

c) Two algorithms A and B solve the same algorithmic problem, A taking n^2 seconds and B taking n days.

(i) Which algorithm is asymptotically preferable?

[3 marks]

(ii) Which algorithm is preferable if n only takes values up to 10,000?

[4 marks]

(iii) How large does n need to be before B takes half the time taken by A?

[4 marks]

TURN OVER

2.

a)

(i) 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]

(ii) *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). 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]

b) Use a simple graphical argument to show that the discrete sum

$$\sum_{i=1}^{n} f(i)$$

is bounded above by the integral

$$\int_{1}^{n+1} f(t)\, dt$$

provided that f(t) is a non-decreasing function.

[6 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.

(i)
```
for (i = 1;  i ≤ n;  i++)
    for (j = 1;  j ≤ n–i;  j++)
        // Do an operation requiring unit time
```

[6 marks]

(ii)
```
for (i = 1;  i ≤ n;  i++)
    for (j = 1;  j ≤ n;  j++)
        for (k = 1;  k ≤ i*j;  k++)
            // Do an operation requiring unit time
```

[8 marks]

CONTINUED

3. Throughout all parts of this question you may assume the variable n is a power of 2 where appropriate

a) Solve the following recurrence relations, simplifying your final answers using O-notation.

(i) $f(0) = 0$

$f(n) = f(n-1) + 2, \quad n > 0$

[4 marks]

(ii) $f(0) = 2$

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

[4 marks]

(iii) $f(0) = 2$

$f(1) = 5$

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

[5 marks]

(iv) $f(0) = 1$

$f(1) = 4$

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

[5 marks]

(v) $f(1) = 3$

$f(2) = 9$

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

[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 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

```
if (n > 0)
    {
    Hanoi(n-1, i, 6-i-j);
    // Move a ring from i to j
    Hanoi(n-1, 6-i-j, j);
    }
```

(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 $2^N - 1$.

[5 marks]

END OF SECTION A

CONTINUED