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

SECTION A

1.

a) Give a mathematical definition of the **order notations**

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

and explain how these mathematical concepts relate to the algorithmic ideas of **worst case** and **best case** analysis. Distinguish carefully between the properties of individual algorithmic solutions to a problem and the properties of the problem itself.

[12 marks]

b) Why is the multiplication of two n-bit integers $\Omega(n)$ in terms of single bit operations?

[4 marks]

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

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

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

[8 marks]

d) Show that

$$\sum_{i=1}^{n} a^i \in \begin{cases} O(1) & \text{if } a < 1 \\ O(n) & \text{if } a = 1 \\ O(a^n) & \text{if } a > 1 \end{cases}$$

[9 marks]

2.

a) Quicksort is widely regarded as one of the best general-purpose sorting algorithms. However there are circumstances under which Quicksort might *not* be a good choice. Give an example of such a situation, and suggest, with a justification of your choice, an alternative sorting algorithm which might be preferable in this case.

[5 marks]

b) It can be shown that all comparison-based sorting algorithms need to use at least some multiple of n log(n) comparison operations in their worst case, where n is the number of items to be sorted.

Countsort is a sorting algorithm which can be shown to use O(n) comparison operations.

(i) Explain why this result does not contradict the earlier statement about comparison-based sorting.

[4 marks]

(ii) When would it be very inefficient to use Countsort? Why? Give an example of a situation where Countsort would be a *good* choice.

[7 marks]

c)

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

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

is bounded above by the integral

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

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

[7 marks]

(ii) Use the result of (i) above to show, starting from the observation that

$$\ln(\, n!\,) \;=\; \sum_{i=1}^{n} \ln(\, i\,)$$

that

$$n! \;\in\; O(\, n(\tfrac{n}{e})^{n}\,)$$

[7 marks]

3.

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

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

[4 marks]

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

[4 marks]

(iii) $f(0) = 1$
$f(1) = 2$
$f(n) = 2f(n-1) - f(n-2), \; n > 1$

[6 marks]

b) The procedure below searches the array A[1..n] for an element equal to k. If k is found, the integer p is set to the array position in which k was discovered, otherwise p is set to zero to indicate ’not found’.

// Sequential search of an array A[1..n] for the element k

```
i = 1;
A[n+1] = large 'sentinel' value;
while (A[i] <> k)
        i = i+1;
if (A[i] == k)
        p = i       // k found at position i
else
        p = 0;      // k not found
```

(i) How many comparisons does this algorithm do in the worst case?

[3 marks]

(ii) Suppose that the array A is **sorted** prior to the search, so that the test ’A[i] <> k?’ can be replaced by ’A[i] < k?’. Show that the **average** number of comparisons for an unsuccessful search is now equal to $\frac{1}{2}(n+2)$.

[4 marks]

[Question 3. cont. over page]

[Question 3. cont.]

c) Assume again that A[1..n] is a sorted array. A more efficient search procedure for such an array is given below:

// Binary search of a sorted array A[b..t] for the element k

```
if (b == t)      // array A has only one element
    {
    if (k == A[b])
        p = b
    else
        p = 0
    }
else      // array A has more than one element
    {
    m = (b+t) div 2;      // integer division to find the median element
    if (k <= A[m])
        // Use binary search on array A[b..m]
    else
        // Use binary search on array A[m+1..t]
    }
```

(i) Write down a recursive formula for B(n), the number of comparisons made in the case of an unsuccessful search of A[1..n] by the above algorithm. Consider the case of both odd and even n.

[6 marks]

(ii) Solve this recurrence to obtain the number of comparisons which must be performed in this case. You may assume that n is a power of 2.

[6 marks]

END OF SECTION A