Marking Scheme: - a) 1.5 marks b) 3 marks c) 1 mark d) 1.5 marks.

```
Input: array A[0..n-1] of n numbers
Output: A is sorted in increasing order.

If n = 2 and A[0] > A[1], then swap (A[0], A[1])
If n > 2 then {
        Stooge-sort(A[0..ceil(2n/3)]) // sort first two-thirds.
        Stooge-sort(A[floor(n/3)..n]) // sort last two-thirds.
        Stooge-sort(A[0..ceil(2n/3)]) // sort first two-thirds again.
}
```

a) Each time the function stooge-sort is called 3 times on 2*n/3 elements and on each call the work done is constant.

$$T(n) = 3T(\frac{2}{3}n) + \Theta(1).$$

b)

$$\begin{aligned} T(n) &= 1 + 3T(\frac{2}{3}n) \\ &= 1 + 3 + 9T(\frac{4}{9}n) \\ & \quad \dots \\ &= 1 + 3 + 3^2 + \dots + 3^{\log_{3/2} n} \\ &= \frac{3^{\log_{3/2} n + 1} - 1}{3 - 1} \\ &= \Theta(3^{\log_{3/2} n}) \\ &= \Theta(3^{(\log_3 n)/(\log_3 3/2)}) \\ &= \Theta(n^{1/(\log_3 3/2)}) \\ &= \Theta(n^{2.71}). \end{aligned}$$

c) Yes.

d) No. Insertion sort complexity is $O(n^2)$ which is equivalent to $O(n^{\log 9/4})$ (log base is 3/2) clearly 9/4<3 hence performance of stooge-sort is worse than insertion sort.