1. Give tight asymptotic bounds for $T(n)$ in each of the following recurrences. Use backward substitution, either version of the Master Theorem, or make a good guess and show that it holds. Assume that $T(n)$ is constant for $n \leq 2$. State your method of solution and if you use either version of the Master Theorem, state the case the recurrence falls into (Case 1, 2, or 3, etc.). Show all your work.

a. $T(n) = T(\frac{9n}{10}) + n$

b. $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$

c. $T(n) = T(n-1) + n$

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

e. $T(n) = 6T(\frac{n}{4}) + n^2$

f. $T(n) = 3T(\frac{n}{2}) + n\log n$
2. For each of the following arrays of keys, give the heap size. That is, if the array is a max-heap at the root and at every other node in the tree, the heap size is the number of items in the array. If the entire array is not a max-heap, give the node at which the max-heap property fails and give the size of the max-heap up until the node where the property fails.

The first two examples are given for you.

<table>
<thead>
<tr>
<th>Array</th>
<th>Heap Property violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 12 16 13 10 8 14</td>
<td>Node 2 is not the root of a max-heap. The heapsize is 1</td>
</tr>
<tr>
<td>25 14 16 13 10 8 12</td>
<td>Each node is the root of a max-heap. The heapsize is 7</td>
</tr>
<tr>
<td>25 14 13 16 10 8 12</td>
<td></td>
</tr>
<tr>
<td>25 14 12 13 10 8 16</td>
<td></td>
</tr>
<tr>
<td>14 13 12 10 8</td>
<td></td>
</tr>
<tr>
<td>14 12 8 10 13</td>
<td></td>
</tr>
<tr>
<td>89 19 40 17 12 10 2</td>
<td></td>
</tr>
</tbody>
</table>

3. What are the minimum and maximum numbers of nodes in a heap of height \( h \)?

4. Order the following functions in terms of their order of growth (i.e., growth rate), from lowest to highest:

\[ \text{nlgn, } n + \text{lgn, } n!, \text{ } n, \text{ } n^2, \text{ } 2^n, \text{ } n^{\frac{1}{2}}, \text{ } 4^n \]

Provide a justification for your answers and indicate which of the functions, if any, grow at the same rate.
Consider the SELECTION-SORT algorithm, given below. Assume the input and output are as specified for the sorting problem on page 16 of our textbook.

\begin{verbatim}
SELECTION-SORT(A):
1.   n = A.length
2.   for j = 1 to n - 1
3.       smallest = j
4.       for i = j + 1 to n
6.               smallest = i
\end{verbatim}

Consider the following loop invariant of the outer for loop (lines 2 through 7).

The algorithm maintains the loop invariant that at the start of each iteration of the outer for loop, the subarray \( A[1..j-1] \) consists of the \( j-1 \) smallest elements in the array \( A[1..n] \) and the subarray \( A[1..j-1] \) is in sorted order.

Show that the loop invariant is true using a formal proof like that shown in class for INSERTION-SORT (prove the invariant holds initially, at every subsequent iteration, and at termination, according to the value of \( j \)).
6. Come up with running time solutions for the following pseudocode samples using either the unrolling method or by multiplying the time taken by each case with nested loops as shown in class. If an inner loop counter \( j \) is dependent on an outer loop counter \( i \), for each value of \( i \), determine how many times the internal loop is repeated and derive a closed-form expression from the summation. If all loops are independent, find the number of times each is executed and multiply the result.

\[
\text{Alg1}(n)
\{ \\
\text{for } (i = 1; i \leq n; i++) \\
\quad \text{for } (j = 1; j \leq i; j++) \\
\quad \quad \text{for } (k = 1; k \leq 100; k++) \\
\quad \quad \quad \text{printf } (i + j + k) \\
\}
\]

\[
\text{Alg2}(n)
\{ \\
\text{for } (i = \frac{n}{2}; i \leq n; i++) \\
\quad \text{for } (j = 1; j \leq n; j = j * 2) \\
\quad \quad \text{for } (k = 1; k \leq n; k = k * 2) \\
\quad \quad \quad \text{printf } (i + j) \\
\}
\]