1. (20 points) Consider the algorithm \texttt{MyAlg(A,n)}.

\begin{verbatim}
Algorithm MyAlg(A,n)
Input: Array A and integer n
if n < 1 return(0)
else
return(temp)
\end{verbatim}

(a) (5 points) Write down recurrence equation which describes the running time of \texttt{MyAlg} as a function of \( n \)

(b) (15 points) Solve this recurrence equation and give the asymptotic (big-O) complexity of the algorithm.

2. (20 points) Consider a hash table of size \( N = 11 \) where we are going to store positive integer values. Suppose our hash function is \( h(k) = k \mod 11 \). Suppose we insert the following values, in the given order: 0, 9, 20, 22, 31, 11, 42

(a) Draw the table that results after inserting, if collisions are handled by linear probing

(b) Draw the table that results after inserting, if collisions are handled by double hashing, with the secondary hash function \( h(k) = 7 \mod 7 \)

3. (10 points) Draw a binary tree where each node stores a character key, and preorder traversal visits nodes in the order CBAFGED, and postorder traversal visits nodes in the order AFBEDGC

4. (10 points) Let \( T \) be an AVL tree storing 256 keys.
   a) (5 points) What is the smallest possible height of this tree?
   b) (5 points) Can this tree have height 17?

5. (10 points) Let \( H \) be a heap of height \( h \). State all the levels of the heap at which the 3\textsuperscript{rd} largest key element can be located? Recall that larger keys correspond to lower priority.

6. (30 points)
   a) (20 points) Write an algorithm which takes as an input the root of a binary tree and computes the number of nodes in the tree which have exactly 1 child. For example, in for the following binary tree, your algorithm should return 1, since only the node \textit{a} has exactly 1 child, namely \textit{c}. You can use all the binary ADT methods discussed in class.
   b) (10 points) Analyze the asymptotic complexity of your algorithm in part (a)

7. Just for fun, not graded.
   Prove that if every internal node in a tree has exactly 3 children, then (number of external nodes) = 2*(number of internal nodes) +1