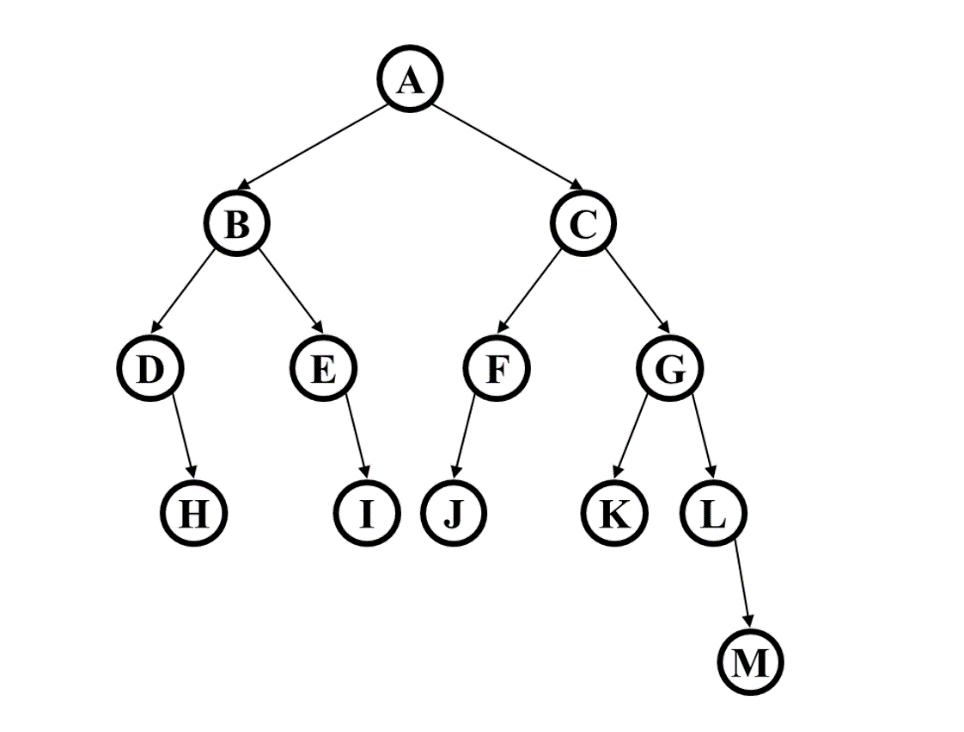
**CS310 Data Structures and Algorithms Spring 2020**

**Homework 3 & 4**

**40 Points**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. (8 Points) Assume this tree is a **Binary Search Tree** even though you cannot see what the keys and values are at the nodes (the letters we write below are just “names” for the nodes for the purpose of answering the questions).



* 1. The value of H is greater than the value of I (True/False)
  2. This Binary Search Tree is complete (True/False)
  3. What is the height of the tree?
  4. What is the maximum number of nodes that could be added to the tree without increasing its height?

1. (4+4 Points) Suppose there is a **Binary Min-Heap** with exactly 4 nodes, containing items with priorities 3, 9, 11, and 15.
   1. Show every possible binary min-heap that could match this description. For each, draw the appropriate tree and the array representation. (You can show just the priorities, not the corresponding items.)
   2. For one of your answers to part (a), show what happens with 4 deleteMin operations. Clearly indicate which heap you are starting with and show the heap after each deleteMin. You can just draw the tree (not the array) after each step.
2. **(8 points)** For each of the following situations, name the best sorting algorithm we studied. (For one or two questions, there may be more than one answer deserving full credit, but you only need to give one answer for each.)

The array is mostly sorted already (a few elements are in the wrong place).

1. You need an O(n log n) sort even in the worst case and you cannot use any extra space except for a few local variables.
2. The data to be sorted is too big to fit in memory, so most of it is on disk.
3. You have many data sets to sort separately, and each one has only around 10 elements.
4. Instead of sorting the entire data set, you only need the k smallest elements where k is an input to the algorithm but is likely to be much smaller than the size of the entire data set.
5. **(5 Points)** Draw the **binary max heap** that results from inserting 6,12,7,10,17,5,15 in that order into an initially empty binary min heap. You do not need to show the array representation of the heap. Draw all intermediate trees.
6. (4+4 Points)

• Describe the most time-efficient way to implement the operations listed below. Assume no duplicate values and that you can implement the operation as a member function of the class – with access to the underlying data structure, including knowing the number of values currently stored (N).

• Then, give the tightest possible upper bound for the worst case running time for each operation in terms of N.

**For any credit, you must explain why it gets this worst case running time.**

1. Given a binary min heap, find which value is the minimum value and delete it.
2. Given a binary search tree, find which value is the minimum value and delete it.