Problem #1: Draw a picture of a binary heap tree using the following array of numbers, {45, 3, 2.1, 9.2, 7, 8.1, 3.6, 24, 95, 1, 6, 0.5, 2, 2.1}. Insert the elements in order, starting with 45, 3, 2.1, etc.

Problem #2: Draw a picture of the binary heap tree in problem #1 after performing four deleteMin() operations. (While percolating you will be forced to choose the smaller of the children. If the children are identical, then choose the leftmost.)

Problem #3: Recall that the insert operator for the binary heap tree uses percolation. At each step, the percolation algorithm requires identifying the parent node. Assume that the binary heap is implemented with a complete tree and an array. Write a Java method that finds the value of a parent for any given node on a binary heap tree.

In other words, write a method that takes the heap and an array index as arguments – this index specifies a node on the tree. Now return the value of the parent that is stored in the heap. You may use the heap implementation that I gave in class, or you may use any other heap implementation.

Problem #4: Assume that there are 3 priority levels with 0 being most important, and 2 being least important. Implement a priority queue by using three regular queues. In other words, write code to implement the priority queue ADT.

Hint: To implement the ADT you must create all of the pertinent methods. You may assume that you already have the code for a Queue (along with all of its
methods – enqueue, dequeue, isEmpty, etc.). However, you must create an appropriate class for the “PriorityQueue” (just like we did for the LinkedList, Stack, Queue, BinaryNode, HashTable, etc.).

Extra Credit (5 points): Rewrite the code for problem #4 so that it can handle any integer number of priority levels.