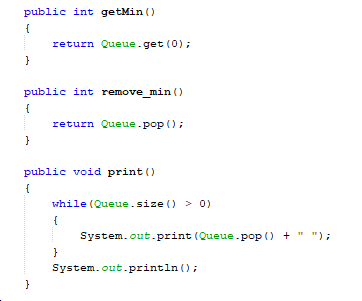
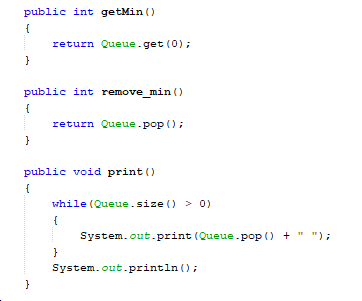
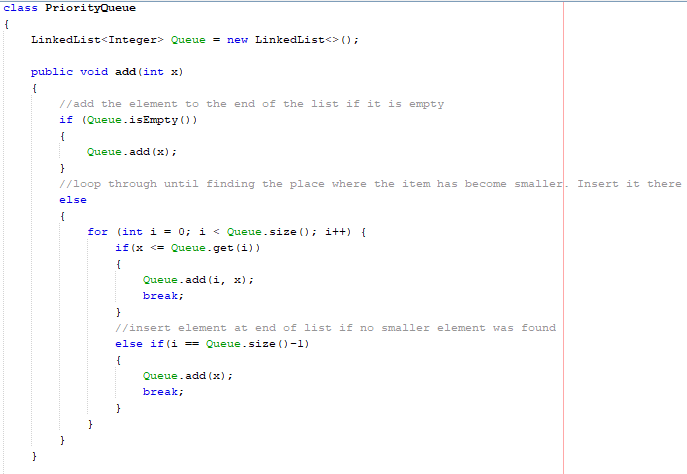
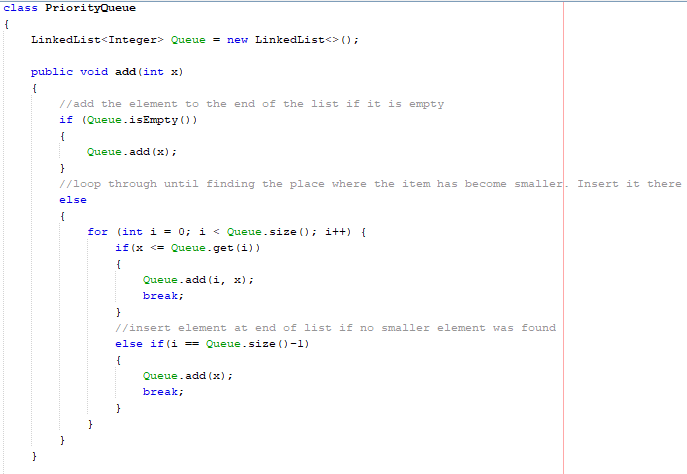
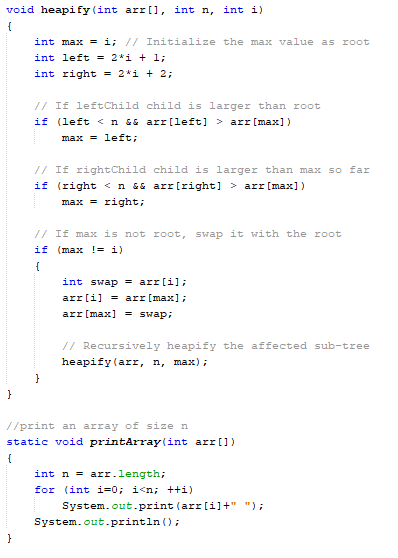
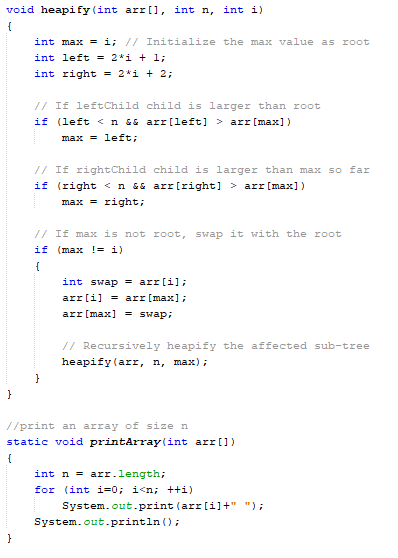
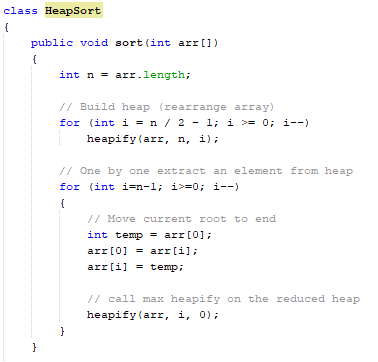
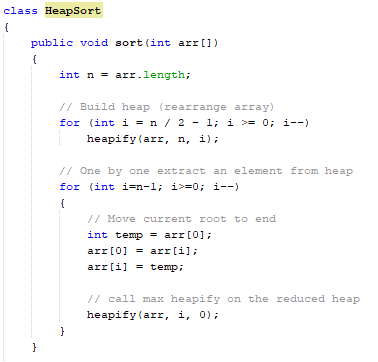
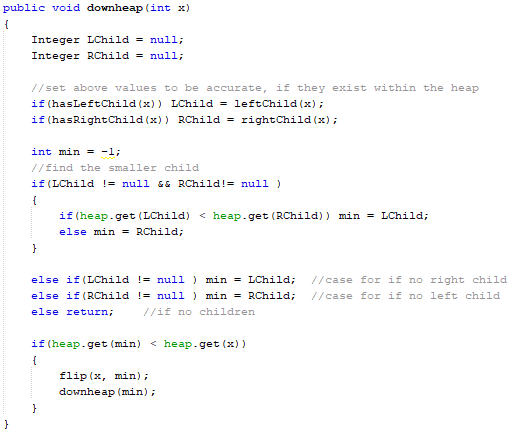
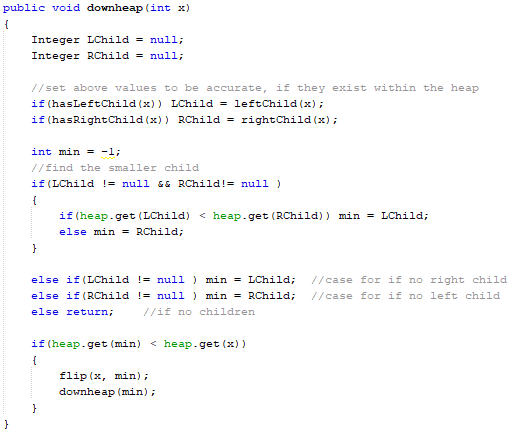
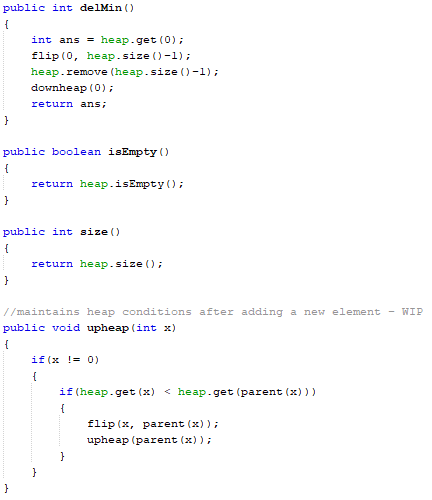
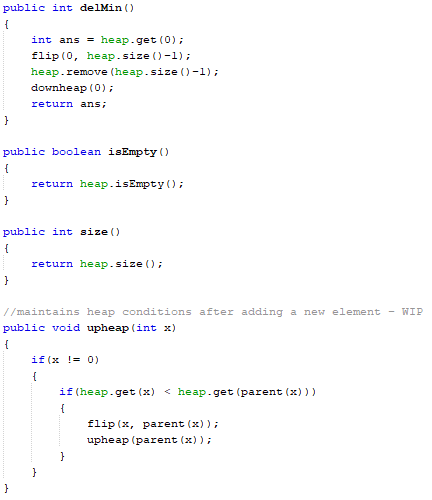
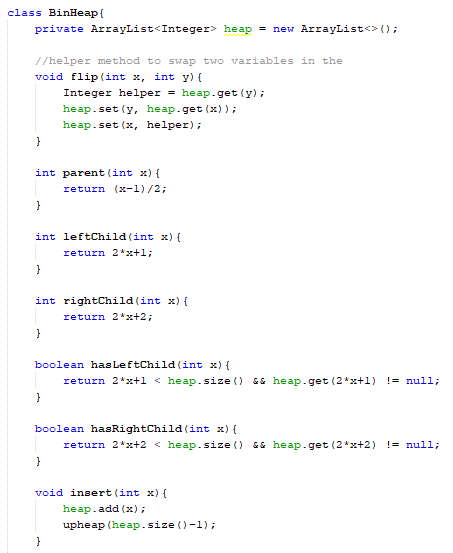
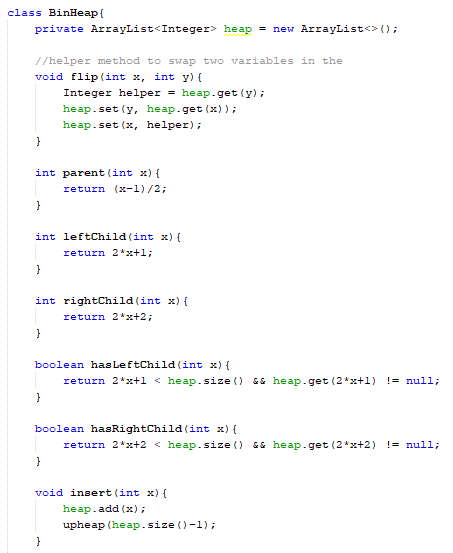
***well-documented summary that includes the annotation/justification of your algorithm and programming solution and the experimental results with the screenshots of your program testing you test. It should be at least 2 pages MS WORD or PDF file, (the more formal the better.) Note that screenshots should be readable to gain the full credits. The instructor will use the screenshots for student’s program demo.***

This document will rationalize the algorithms and functions used in my HW6 for CSC 310

1. The priority Queue class has 4 main methods. The first is the add method, which incorporates the variables into an ever growing Priority Queue. It adds them to the queue based on where they fit in the sorted collection of elements. The next two methods are fairly self evident and simple, they are getMin and remove\_min, which return or remove and return the minimum value from the Priority queue, respectively. The final method for the Priority queue class is the print method, which simply outputs the contents of the sorted queue to the user, while simultaneously emptying out the queue.
2. The heapsort class only has 3 methods, and functions as a toolbox for an in-place sort method. The class depends on the array representation of a heap. The first method is the in-place sort method, which recursively ”heapifys” smaller and smaller arrays. The heapify method compares child nodes to their parent, and sets the value stored within to be the highest value if it is larger than the pre-existing max value. It then swaps the max value(max) with the first value in the array, which represents the root(i), assuming the two are not the same. Then, the effected subtree is recursively heapified, to ensure that the now sorted portion of the heap is untouched and that the heap continues to follow the rules that make it a heap. The print array method simply prints out all the elements stored within the heap.
3. The final, binheap portion of the assignment has several methods. Flip swaps given elements, parent, leftchild, and rightchild return the value of a node’s children based on array representation of a tree/heap, while hasRightChild and hasLeftChild simply determine if there exists a right or left child. Insert adds an element to the heap, and then calls upheap to ensure it winds up in the right place in the heap. Upheap functions by finding if a given node is less than it’s parent and swaps the two if it is. After the swap, it then calls upheap on the new position of the original value. This continues until the parent node is no longer greater than it’s child.  
   DelMin swaps the smallest value in the heap with the last value in the heap, before removing it from the heap. This is necessary because only the last value in a heap can be removed. The method downheap is then called with respect to the remaining swapped variable, to ensure that the heap conditions are still met. Downheap determines the left and right children of the root, which delMin had changed prior to calling downheap. It then finds the smaller child, and swaps the root and that child before calling downheap on the new location of the changed root.