Problem 1

Several functions occur in the implementation of Problem 1. These include insert, search, delete, and sort. Insertion alone would be O(1), as all that really happens is hepe[k] = kelem; with the **while** (kelem < hepe[(k-1)/childct]&&k > 0) taking roughly O(k) time, where k is the size of the heap.

An additional level of complexity is added if the heap has to be doubled – that is, the array size is exceeded and its elements have to be copied to a new one. This depends on O(k) as well, as only k elements get copied and array instantiation is O(1). Thus, this operation in total is O(2k) which is roughly O(k).

Search and delete are also O(k), as only k elements at worst need to be traversed to find the needed heap entry.

The heapsort depends on the tree height. This is log-base-n of k, where n is the number of children allowed and k is the number of elements in the heap. At worst, the tree must be run up and down k times, along its lognk height. This is klogk, which is O(nlogn).