Programming Assignment - 8
Quicksort and Priority Queue

1) Priority Queues

Suppose that the sequence M Y P R I O * R * * I * T * Y * * * Q U E * * * U* E* (where a letter means *insert* and an *asterisk* means remove the maximum) is applied to an initially empty priority queue. **Give the sequence of heaps** and the sequence letters returned by the *remove the maximum* operations.

2) Quicksort – Optimized version of 3-Way Quicksort

Implement an optimized version 3-Way quicksort algorithm with the following changes:

a) Pivot: (i) First element (or any random element) and (ii) median of a[left], a[center], and a[right] of the subarray (or any three random elements in the subarray)

b) Cutoff to insertion sort for subarrays with less than M elements from 0 to 30.

You need to add the following two methods:

a) getPivot(a, lo, hi) method that returns the pivot element based on your chosen strategy in the subarray a[lo..hi]

b) insertionSort(a, lo, hi) that sorts the subarray a[lo..hi]

Empirically determine the value M for which value of quicksort runs fasted in your environment to sort random arrays of N doubles, for N = 10^3, 10^4, 10^5, and 10^6. Plot the average running times for M from 0 to 30.

3) Ternary heapsort

Implement a version of heapsort based on complete **ternary ary** heap as similar to the binary heap described in the text. Test your implementation using 100 randomly ordered distinct keys.