Median Queue

A *median queue* is an abstract data type for a collection that supports the following operations:

- `insert`(Q, x) – inserts an element x into the queue
- `removeMedian`(Q) – removes and returns the *median* element from the queue

The median is the "middle" element of the collection when the elements are arranged from smallest to largest. If we insert the numbers $8, 3, 1, 5$, and 7 into a median queue, then:

- the first call to `removeMedian` will give 5
- the second call to `removeMedian` will give 3
- the third call to `removeMedian` will give 7
- the fourth call to `removeMedian` will give 1
- the fifth call to `removeMedian` will give 8

Note that we choose the smaller of the two middle elements when the queue contains an even number of elements, as shown in the example above.

(a) If we insert the numbers $4, 1, 8, 3, 2$ into a median queue, what will the first three calls to `removeMedian` give? Provide the answer as a comma-separated list of numbers.

(b) Explain *briefly* how you would represent a median queue.

(c) Explain how you would implement the operations `insert` and `removeMedian`. You can use pseudocode as part of your explanation, but it is not a requirement. All data structures mentioned in the curriculum are available to you.

(d) Based on your explanation in (c), what can you say about the runtime complexity of `insert` and `removeMedian`?

(e) If we take n elements from a median queue using `removeMedian`, and insert them into a binary search tree, one by one, what will the resulting binary search tree look like? Explain briefly.