Exercise 1 ....

Min/max priority queue. Design a data type that supports the following operations: insert, delete the maximum, and delete the minimum (all in logarithmic time); and find the maximum and find the minimum (both in constant time). Hint: Use two heaps.

Exercise 2 ....

Dynamic-median finding. Design a data type that supports insert in logarithmic time, find the median in constant time, and remove the median in logarithmic time.

Hint: Keep the median key in `v`; use a max-oriented heap for keys less than the key of `v`; use a min-oriented heap for keys greater than the key of `v`. To insert, add the new key into the appropriate heap, replace `v` with the key extracted from that heap.

Exercise 3 ....

Fast insert. Develop a compare-based implementation of the MinPQ API such that insert uses log log N compares and delete the minimum uses 2 log N compares. Hint : Use binary search on parent pointers to find the ancestor in swim().