Problem 4-1. Extreme FIFO Queues [25 points]

Design a data structure that maintains a FIFO queue of integers, supporting operations ENQUEUE, DEQUEUE, and FIND-MIN, each in $O(1)$ amortized time. In other words, any sequence of $m$ operations should take time $O(m)$. You may assume that, in any execution, all the items that get enqueued are distinct.

(a) [5 points] Describe your data structure. Include clear invariants describing its key properties. *Hint:* Use an actual queue plus auxiliary data structure(s) for bookkeeping.

(b) [5 points] Describe carefully, in words or pseudo-code, your ENQUEUE, DEQUEUE and FIND-MIN procedures.

(c) [5 points] Prove that your operations give the right answers. *Hint:* You may want to prove that their correctness follows from your data structure invariants. In that case you should also sketch arguments for why the invariants hold.

(d) [10 points] Analyze the time complexity: the worst-case cost for each operation, and the amortized cost of any sequence of $m$ operations.
Problem 4-2. Quicksort Analysis [25 points]

In this problem, we will analyze the time complexity of QUICKSORT in terms of error probabilities, rather than in terms of expectation. Suppose the array to be sorted is \( A[1..n] \), and write \( x_i \) for the element that starts in array location \( A[i] \) (before QUICKSORT is called). Assume that all the \( x_i \) values are distinct.

In solving this problem, it will be useful to recall a claim from lecture. Here it is, slightly restated:

**Claim:** Let \( c > 1 \) be a real constant, and let \( \alpha \) be a positive integer. Then, with probability at least
\[
1 - \frac{1}{n^\alpha},
\]
\( 3(\alpha + c) \log n \) tosses of a fair coin produce at least \( c \log n \) heads.

**Note:** High probability bounds, and this Claim, will be covered in Tuesday’s lecture.

(a) [5 points] Consider a particular element \( x_i \). Consider a recursive call of QUICKSORT on subarray \( A[p..p+m−1] \) of size \( m \geq 2 \) which includes element \( x_i \). Prove that, with probability at least \( \frac{1}{2} \), either this call to QUICKSORT chooses \( x_i \) as the pivot element, or the next recursive call to QUICKSORT containing \( x_i \) involves a subarray of size at most \( \frac{3}{4} m \).

(b) [9 points] Consider a particular element \( x_i \). Prove that, with probability at least \( 1 - \frac{1}{n^\alpha} \),

the total number of times the algorithm compares \( x_i \) with pivots is at most \( d \log n \), for a particular constant \( d \). Give a value for \( d \) explicitly.

(c) [6 points] Now consider all of the elements \( x_1, x_2, \ldots, x_n \). Apply your result from part (b) to prove that, with probability at least \( 1 - \frac{1}{n^\alpha} \), the total number of comparisons made by QUICKSORT on the given array input is at most \( d' n \log n \), for a particular constant \( d' \). Give a value for \( d' \) explicitly. **Hint:** The Union Bound may be useful for your analysis.

(d) [5 points] Generalize your results above to obtain a bound on the number of comparisons made by QUICKSORT that holds with probability \( 1 - \frac{1}{n^\alpha} \), for any positive integer \( \alpha \), rather than just probability \( 1 - \frac{1}{n} \) (i.e., \( \alpha = 1 \)).