1. A colleague claims to be able to implement a comparison-based priority queue that only uses \( \log_2(\sqrt{n}) \) comparisons for insertion, and \( \sqrt{\log_2 n} \) comparisons for \texttt{DeleteMin}. How can you politely explain to your colleague that either (1) the claim that the algorithm is comparison-based is false, (2) the claim that the algorithm is correct is false, or (3) the claim about the algorithm’s running time is false.

2. Show that the total number of recursive invocations of the quicksort algorithm discussed in class is \( O(n) \) in the worst case.

3. The quicksort algorithm presented in class contains two recursive calls to itself. The second recursive call is not really necessary, and can be easily removed using a technique called \texttt{tail recursion}, as illustrated below:

   ```
   procedure QuickSort\(^\prime\)(A, p, r)
   while p < r do
     q ← Partition(A, p, r)
     QuickSort\(^\prime\)(A, p, q − 1)
     p ← q + 1
   
   Here, the procedure \texttt{Partition} is the same as that described in class, and may be implemented either with a deterministic choice of pivot index, or a random choice of pivot index.

   (a) Briefly argue that \texttt{QuickSort\(^\prime\)(A, 1, n)} correctly sorts the array \( A[1\ldots n] \), using exactly the same number of comparisons in the worst and average case as the procedure \texttt{QuickSort} presented in class.

   (b) Show that the height of the recursion tree (i.e., the “stack depth”) of \texttt{QuickSort\(^\prime\)} is \( \Theta(n) \) in the worst case.

   (c) Modify the code for \texttt{QuickSort\(^\prime\)} so that the height of the recursion tree is \( O(\log n) \) in the worst case, but without affecting the number of comparisons in the worst or average case.

4. Consider the quicksort algorithm discussed in class using the randomized partition procedure. Suppose that we view random numbers as a scarce resource, and that we charge \( \log_2(r − p + 1) \) “random units” to generate a random pivot index in the range \( p \ldots r \). The intuition here is that we charge one random unit for every bit of the length of interval from which a random number is drawn.

   (a) Show that in the worst case (choosing the input and pivot indices in a malicious way), the total number of random units used by quicksort on inputs of length \( n \) is \( \Theta(n \log n) \).

   (b) Show that the expected number of random units used is \( O(n) \). Hint: this is a bit tricky; modify the analysis of quicksort presented in class, replacing the “cost function” \( n − 1 \) by the “cost function” \( \log_2 n \); also, the following (easy) inequality may be useful: \( \log_2 n \leq \log_2(n − 1) + 2/(n \ln(2)) \) for \( n \geq 2 \).