1. Prove or disprove each of the following conjectures.
   a. $f(n) = O(g(n))$ implies $g(n) = O(f(n))$.
   b. $f(n) = O((f(n))^2)$
   c. $\sum_{i=1}^{n} \frac{1}{i^2} = O(1)$.

2. Let $T(n)$ be the worst-case time complexity $T(n)$ of the Heapsort algorithm $Heapsort(A)$ given in Chapter 6 of the CLRS textbook ($n$ is the length of the array $A$). As discussed in the book, $T(n) = O(n \log n)$. Is $T(n) = \Omega(n \log n)$? Prove your answer.

   A $d$-ary heap is like a binary heap, but (with one possible exception) non-leaf nodes have $d$ children instead of 2 children.
   (a) How would you represent a $d$-ary heap in an array?
   (b) What is the height of a $d$-ary heap of $n$ elements in terms of $n$ and $d$?
   (c) Give an efficient implementation of ExtractMax in a $d$-ary max heap. Analyze its running time in terms of $d$ and $n$.
   (d) Give an efficient implementation of Insert in a $d$-ary max heap. Analyze its running time in terms of $d$ and $n$.
   (e) Give an efficient implementation of IncreaseKey($A, i, k$), which flags an error if $k < A[i]$, but otherwise sets $A[i] = k$ and then updates the $d$-ary maxheap structure appropriately. Analyze its running time in terms of $d$ and $n$.

4. Give an algorithm that uses one of the data structures that we have studied so far to perform the following. The input consists of $k$ sorted lists $L_1, \ldots, L_k$, each one containing a list of $n/k$ integers in increasing order. The algorithm should output a single list $L$ that contains the $n$ integers in $A_1, \ldots, A_k$, sorted in increasing order.
   a. Give a simple algorithm for solving the above problem with worst-case time complexity $O(n \log k)$. Explain why it works. Give a clear and concise description of your algorithm in English. Do not use pseudocode.
   b. Explain why your algorithm’s worst-case time complexity is $\Omega(n \log k)$.