Fall 2018, CMPSC 465: Take-home Exam 1.
No collaboration, no discussion with others permitted. You may refer to your notes, material on the Canvas class site, and the three class textbooks, but you are not permitted to access any other resource. Please read the statement on academic integrity in the syllabus.
Solutions must be typed and electronically prepared (use LaTeX or Word). Scans and photographs won't be graded. Please submit a single PDF file on Canvas.
The submission deadline is **9 pm on October 17th**. It will be strictly enforced.
The exam is for 100 points, and each problem is for 20 points.

---

1. Prove or disprove:
   - For all functions $f(n)$ and $g(n)$, we must have that either $f(n) = \mathrm{O}(g(n))$ or $g(n) = \mathrm{O}(f(n))$.
   - If $f(n) = \mathrm{O}(g(n))$, then $2^{f(n)} = \mathrm{O}(2^{g(n)})$.
2. Given two sorted sequences of length m and n, respectively, and assuming $n > m$, design an efficient $\mathrm{O}(\log n)$ algorithm to find the kth smallest element in the merged sequence (of length $m+n$). Note that you should not explicitly merge the two sequences, because merging will take $O(m+n)$ time.
3. You are given a book and a short text message. You are asked to cut letters from the book to compose the message. Give an efficient algorithm to determine if the message can be composed using this scheme (i.e., do all the letters in the message occur in the book?). Clearly state your assumptions, define variables, and use data structures as necessary.
4. A is an array of length n with distinct integers. The first k elements of A are in increasing order, and the next $n-k$ elements are in decreasing order. We are not given the value of k. Give pseudocode for a logarithmic-time divide-and-conquer algorithm to locate the maximum element of A.
5. You are told that A is a 20×20 matrix with distinct positive integers. The values in each row are sorted and in increasing order (left to right). The values in each column are also in sorted, increasing order (top to bottom). Suppose you cannot access A directly, and inspecting the value at even a single location is very expensive. Your goal is to tell whether a value k appears in the matrix A. Give an algorithm that is guaranteed to tell if k occurs in the matrix in at most 40 location accesses (i.e., accessing just 10% of the matrix entries). You may use any auxiliary data structures, but the main goal is to minimize accesses to A.