Please do all problems; we will grade a subset of the assigned problems (same subset for everyone). Please follow all of the instructions given in the handout for Homework 1.

1. (16 points) Problem 13.2, p. 332-333. (Skip part (e) as it is symmetric to part (b).)
Supplement your answer with short code fragments, as appropriate.

2. (12 points) Let $A$ be an array of distinct integers. Elements $A[i]$ and $A[j]$ form an inversion if $i < j$ and $A[i] > A[j]$. We would like to count the number of inversions in $A$. For instance $A = 3, 1, 4, 2, 6$ has three inversions, i.e., (3,1), (3,2), and (4,2).
Give an $O(n \log n)$-time algorithm for this problem which uses an order-statistic tree as a black-box. (There are other ways to solve this problem within this time bound, but to earn any credit your solution should be based on an order-statistic tree.)
Your answer should include (a) a discussion of the main ideas underlying your solution from which the correctness of your method is clear, (b) pseudocode, and (c) an analysis of the running time.

3. (12 points) This problem explores a solution to the dynamic 1-dimensional closest pair (1D-CP) problem, where the goal is to maintain the closest pair amongst points on the real line, as points are inserted/deleted.
Let $S$ be a set of reals that is subject to insertions and deletions. We wish to maintain $S$ under these updates and to also support the following operation—all done efficiently:

- **1D-CP($S$)**: Return the absolute value of the smallest difference between two numbers in $S$. For example, if $S = \{7.1, 3.2, 1.4, 6.5, 10.2\}$, then 1D-CP($S$) returns $|\ 7.1 - 6.5\ | = 0.6$. If $|S| \leq 1$, then 1D-CP($S$) returns $\infty$.

The target time bounds are $O(\log n)$ for insertions and deletions, where $n$ is the current size of $S$, and $O(1)$ for 1D-CP.
Show how to use an augmented red-black tree to solve this problem. Be sure to state clearly the auxiliary information you store at each node. (You may need more than one auxiliary field at each node.) You must use the GAT; proving the result from scratch will earn no credit.
Your answer should include (a) a discussion of the main ideas underlying your solution from which the correctness of your method is clear, (b) a brief discussion (in words) of how the different operations are done; no pseudocode is required, and (c) an analysis of the running time.

Over $\Rightarrow$
4. (15 points) Let $R$ be a set of red points and $B$ a set of blue points in the plane, where $|R| + |B| = n$. At time $t = 0$ the red points start moving horizontally rightwards and the blue points start moving vertically upwards, all at a speed of 1 mph. We would like to know whether some red point and some blue point ever collide in the future, i.e., at some time $t \geq 0$. (The output should simply be a “yes” or a “no”.)

(a) At any time $t$, let the position of a red point, $r$, be $(x_r(t), y_r(t))$ and that of a blue point, $b$, be $(x_b(t), y_b(t))$. Prove that $r$ and $b$ collide at some time $t \geq 0$ if and only if $x_r(0) + y_r(0) = x_b(0) + y_b(0)$ and $x_r(0) \leq x_b(0)$.

(b) Based on part (a), design a sweepline algorithm that uses a suitable data structure to solve this collision detection problem in $O(n \log n)$ time. Your answer should include (a) a brief description of the key ideas from which the correctness of your method should be clear, (b) pseudocode, and (c) an analysis of the running time.

5. (12 points) Suppose that you wish to do two types of operations on an infinite binary counter: INCREMENT, which is as defined in class, and RESET, which resets all bits in the counter to zero. Explain, in words, how such a counter could be implemented so that any sequence of $n$ INCREMENT and RESET operations on an initially-zero counter takes $O(n)$ time. Use the accounting (i.e., credits-based) method to do the analysis. State clearly the invariant you use and the amortized cost that you assign to each operation. (Assume that it takes constant time to examine or to modify a bit.)

Hint: Consider keeping a pointer to the high-order 1-bit.

6. (12 points) This problem assumes familiarity with Ch. 6. In that chapter, it is shown that a max-heap, $A$, on $n$ keys can be built in $O(n)$ time (See Sec. 6.3). Derive this result using the potential method of amortized analysis. State clearly your potential function.

Hint: Try to relate your potential function to the heights of the maximal max-heaps present just before the next HEAPIFY operation is done. For instance, in Fig. 6.3(d), just before HEAPIFY is done from the node numbered 2 (shown dark gray), the maximal max-heaps are the ones with roots numbered 3, 4, and 5. They are maximal in the sense that none of them is contained in a larger max-heap at that moment.

This problem illustrates two aspects of amortized analysis: i.e., it often leads to simpler proofs and it “forces” one to really understand how a data structure works.