This exam is worth 100pts and you have 55 minutes to complete it. You can implement the solutions in the language of your choice (or pseudocode that is exactly as complex as the equivalent C or Java code). Write your name above and on each answer sheet.

You can use the following method that has complexity $O(n \log n)$: sort(list1, list2) which sorts 2 different lists based on the sorted order of the first list. You can also assume the presence of methods max and min which return the larger (smaller) of two arguments.

1. [33] You just ran a marathon and are quite thirsty. You need to maximize your electrolyte intake. You have a set of fluids that each has a quantity and amount of electrolytes associated with it. Write a greedy algorithm (which is $O(n \log n)$ to give the maximum electrolyte count you can get. Example: amounts=[220, 50,300,115,400] (in ml), electrolytes=[5,11,10,4,12] (per ml) and thirst=600 (in ml); return 6850 (you should drink 50ml of liquid 2, 150ml of liquid 3, and 400ml of liquid 5, so you return (50 * 11 + 150 * 10 + 400 * 12)). [My solution was 7 lines.] int getElectrolyteCount(int n, int thirst, int * amounts, int * electrolytes)

2. [34] Write a method that uses a divide and conquer approach to calculate, in $O(n)$ time, the number of scores in a list that fall into a given range of percentages. For example, given the list [50, 40, 70, 80, 100, 60, 20, 30, 50], and the percentages 60 and 80%, your method should return 3, because 70, 60, and 80 fall between those percentages. Assume that the two ends of the range are elements in the list. If you need to find a pivot, don’t worry about the quality of that pivot. Use a helper method. [My solution was 33 lines.] int getCountInRange(int n, int * elements, int start, int end) /*This problem was worded incorrectly. It was intended that it be solved using the selection algorithm, but, given the way it was stated, you could also just do a linear traversal of the list */

3. [33] Given a list of numbers, write a method (using dynamic programming) to find the largest sum of any subset, subject to the constraint that the subset may not contain two numbers that were adjacent in the original set. For example, given the list [5, 7, 8, 4, 3, 2, 6], you would return 22 (i.e., 6 + 3 + 8 + 5). [My solution was 7 lines.] int getLargestSum(int n, int * elements)