Counting Sort (2 points)

1. [2 points] Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a \dots b]$ in $O(1)$ time. Your algorithm should use $\Theta(n+k)$ preprocessing time.

> **Solution:** The algorithm will begin by preprocessing exactly as COUNTING-SORT (page 195 - initializing a zero-array, C, of size k and for each number increase the count of that number in the array and finally subtract $C[i-1]$ from $C[i]$) does in lines 1 through 9, so that $C[i]$ contains the number of elements less than or equal to i in the array. When queried about how many integers fall into a range $[a..b]$, simply compute $C[b] - C[a-1]$. This takes $O(1)$ times and yields the desired output.

Bucket Sort (5 points)

2. [2 points] Using Figure 8.4 as a model, illustrate the operation of BUCKET-SORT on the array $A = \langle .79, .13, .16, .64, .39, .20, .89, .53, .71, 42 \rangle$.

> **Solution:** The sublists formed are $\langle .13, .16 \rangle, \langle .20 \rangle, \langle .39 \rangle, \langle .42 \rangle, \langle .53 \rangle, \langle .64 \rangle, \langle .64 \rangle, \langle .71, .79 \rangle,$ $\langle .89 \rangle$. Putting them together, we get $\langle .13, .16, .20, .39, .42, .53, .64, .71, .79, .89 \rangle$

3. [3 points] Explain why the worst-case running time for bucket sort is $\Theta\left(n^2\right)$. What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time $O(n \lg n)$?

> **Solution:** In the worst case, we could have a bucket which contains all n values of the array. Since insertion sort has worst case running time $O\left(n^2\right)$, so does Bucket sort. We can avoid this by using merge sort to sort each bucket instead, which has worst case running time $O(n \lg n)$.

Radix Sort (3 points)

4. [3 points] Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

> **Solution:** First run through the list of integers and convert each one to base n, then radix sort them. Each number will have at most $\log_n\left(n^3\right) = 3$ digits so there will only need to be 3 passes. For each pass, there are n possible values which can be taken on, so we can use counting sort to sort each digit in $O(n)$ time.