CSI 503 – Data Structures and Algorithms
Pseudocode for Counting Sort, Radix Sort and Bucket Sort

Handout 9.1

(a) Pseudocode for Counting Sort:

```
Counting_Sort (A, B, k)

  // A [1 .. n] -- Input array to be sorted. (Each value in A
  //               is an integer the range 1 through k.)
  // B [1 .. n] -- Sorted output array.
  // C [1 .. k] -- Array of counters.

  1. for i = 1  to k do  // Initialize counters.
       C[i] = 0

  // At the end of the following for loop, C[i] stores
  // the number of keys equal to i.

  2. for j = 1 to n do
       C[A[j]] = C[A[j]]+1

  // At the end of the following for loop, C[i] stores
  // the number of keys less than or equal to i.

  3. for i = 2 to k do
       C[i] = C[i] + C[i-1]

  // The following loop uses "downto" to ensure a stable sort.

  4. for j = n downto 1 do

       // Place A[j] in its correct position.

       4.1  B[C[A[j]]] = A[j]

      // If there is another key equal to A[j], it will
      // go before the current A[j]. (Needed for stable sort.)

       4.2  C[A[j]] = C[A[j]] - 1
```

(over)

(b) Pseudocode for Radix Sort:

```
Radix-Sort(A, d)

  // Each key in A[1..n] is a d-digit integer. (Digits are
  // numbered 1 to d from right to left.)

  1. for i = 1 to d do
       Use a stable sorting algorithm to sort A on digit i.
```

(b) Pseudocode for Bucket Sort:

```
Bucket-Sort(A)

  // Each element of A[1..n] is a real number in the interval [0,1).
  // B[0 .. n-1] is an array of pointers.

  1.  for i = 1 to n do
        Insert A[i] into the list pointed to by B[floor(n*A[i])].

  2.  for i = 0 to n-1 do
        Sort list B[i] into increasing order using Insertion-Sort.

  3.  Concatenate lists B[0], B[1], ..., B[n-1] together in order.
```
