# CSCI 241 Data Structures
# Winter 2018
# Assignment 1

## Submitting Your Work

**Assignment 1 Submission** item on the course web site. You must submit your assignment by Midnight, February 7.

Your assignments will be evaluated on correct functionality and conformance to the coding standards described at the end of this assignment specification.

## The Task

Your task is to implement the radix sort algorithm to sort an array of positive integers into ascending order (smallest to largest). The radix sort method must use the counting sort algorithm to sort the numbers into ascending number for each digit of the unsorted numbers.

For example, if the unsorted numbers are: 836, 219, 723, 515, 934, 755, the radix sort will proceed as shown below:

| **Unsorted** | **Sorted on the rightmost (least-significant) digit** | **Sorted on the middle digit** | **Sorted on the leftmost (most-significant) digit** |
|---|---|---|---|
| 836 | 723 | 515 | 219 |
| 219 | 934 | 219 | 515 |
| 723 | 515 | 723 | 723 |
| 515 | 755 | 934 | 755 |
| 934 | 836 | 836 | 836 |
| 755 | 219 | 755 | 934 |

Since each step in the radix sort involves sorting the numbers based on just one digit, the range of values to be used as the basis for the sort is 0 through 9. The counting sort is ideal for this purpose.

## Provided Java Program

The Java program Assignment1.java is provided for your use. It will also be used to test your sort methods when the assignment is graded. Once compiled, the Assignment1 program may be run by the command

java Assignment1 *size min max*

Where *size* is the size of the array to be sorted, *min* is the minimum value of integers in the array and *max* is the maximum value of integers in the array. For example:

java Assignment1 10000 100000 999999

Will generate an array of 10,000 integers, each integer in the range 100,000 to 999,999 and then call your sort method to sort the array.

The Assignment1.java program does the following things:

1. Checks that the command-line arguments are integers and contain sensible values (size > 0, min ≥ 0, min < max).

   if (argErrors(args, argvals)) return;

2. Generates the array of random integers in the specified range.

   int[] list = generateList(argvals[0], argvals[1], argvals[2]);

3. Creates an instance of the RadixSorter class, which you must provide.

   RadixSorter sorter = new RadixSorter();

4. Calls the sort() instance method of the RadixSorter object to sort the array of integers.

   sorter.sort(list);

5. Checks whether the array was correctly sorted.

   ```
   for (int i = 1, i < list.length; i++)
      if (list[i] < list[i-1])
         System.out.println("Error: " + list[i-1] + ">" + list[i]);
   ```

## What You Need to Do

You must write two Java classes:

1. RadixSorter.java, which contains the instance method

   public void sort(int[] list);

   which uses the radix sort algorithm to sort the integer array into ascending order.

2. CountingSorter.java, which contains a constructor:

   public CountingSorter(int biggest);

which creates an instance of the class and sets the upper end of the range of integers, 0 .. *biggest*, that the counting sort algorithm will need to deal with. Your radix sort should create an instance of the CountingSorter class:

CountingSorter sorter = new CountingSorter(9);

to sort the numbers based on digit values in the range 0 .. 9.

The CountingSorter class must also contain an instance method

public void sort(int[] value, int digit[]);

where *value* is the array of integers to be sorted and *digit* is the array of digits from those numbers, to be used as the basis for the sort.

Your radix sort should invoke the counting sort method:

sorter.sort(value, digits)

## Coding Standards

1. Use meaningful names for variables that give the reader a clue as to the purpose of the thing being named.
2. Use comments at the start of the program to identify the purpose of the program, the author and the date written.
3. Use comments at the start of each method to describe the purpose of the method, the purpose of each parameter to the method, and the return value from the method (if any).
4. Use comments at the start of each section of the program to explain what that part of the program does.
5. Use consistent indentation.