1. ***(Bucket Sort)*** A bucket sort begins with a one-dimensional array of positive integers to be sorted and a two-dimensional array of integers with rows indexed from 0 to 9 and columns indexed from 0 to $n - 1$, where n is the number of values to be sorted. Each row of the two-dimensional array is referred to as a *bucket*. Write a class named Bucket Sort containing a method called sort that operates as follows:

a. Place each value of the one-dimensional array into a row of the bucket array, based on the value's "ones" (rightmost) digit. For example, 97 is placed in row 7, 3 is placed in row 3 and 100 is placed in row 0. This procedure is called a *distribution pass*.

b. Loop through the bucket array row by row, and copy the values back to the original array. This procedure is called a *gathering pass*. The new order of the preceding values in the one-dimensional array is 100, 3 and 97.

c. Repeat this process for each subsequent digit position (tens, hundreds, thousands, etc.). On the second (tens digit) pass, 100 is placed in row 0, 3 is placed in row 0 (because 3 has no tens digit) and 97 is placed in row 9. After the gathering pass, the order of the values in the one-dimensional array is 100, 3 and 97. On the third (hundreds digit) pass, 100 is placed in row 1, 3 is placed in row 0 and 97 is placed in row 0 (after the 3).

N.B After this last gathering pass, the original array is in sorted order.

2. Modify the insertion sort so that it sorts the array indirectly. This requires a separate *index array* whose values are the indexes of the actual data elements. The indirect sort rearranges the index array, leaving the data array unchanged.

3. When implementing Insertion Sort, a binary search could be used to locate the position within the first (i – 1) elements of the array into which element i should be inserted. Write java code to implement this idea.

4. One possible improvement for Bubble Sort would be to add a flag variable and a test that determines if an exchange was made during the current iteration. If no exchange was made, then the list is sorted and so the algorithm can stop early. This makes the best case performance become O(n) (because if the list is already sorted, then no iterations will take place on the first pass, and the sort will stop right there). Modify the Bubble Sort implementation to add this flag and test.