Project 1: This project 1 is an extension of your project 0.

It is a sorting algorithm, formally called the bucket-sort.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Language: C++ and Java

Due Date: C++ Soft copy: 2/7/2017 Tuesday before midnight

Due Date: Java Soft copy: 2/8/2017 Wednesday before midnight

Due Date: C++ Hard copy: 2/9/2017 Thursday in class

Due Date: Java Hard copy: 2/9/2017 Thursday in class

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. Input: a text file contains a list of numbers (positive integers).

The name of the input file should be given in argv[1] for C++ and args[0] for Java

II. Output: a text file contains sorted input numbers.

The name of the output file should be given in argv[2] for C++ and args[1] for Java

III. Data structure:

- A bucketSort class

- BucketAry[ ] // an 1-D integer array (dynamically allocated)

- int min

- int max

- int lastIndex // set to max - min + 1, this is the last index used in BucketAry

methods:

- findMinMax // this method read the input file

// to determine the min and max integers in the input file

// it will inform the user if there is a negative number in the input file

// and exits the program

- constructor (min, max) // the constructor dynamically allocates

// the BucketAry size of max - min +1 and set lastIndex to max-min+1

- getIndex (data) // the method returns data - min,

// the adjusted index of the BucketAry for the data

- printSortData // the method prints the sorted input integers from BucketAry

// if bucketAry[i] is zero, print nothing

// if bucketAry[i] > zero, print as many i as the value of bucketAry[i].

// i.e., if bucketAry[5] is 3, it means there is three 5 in the input file, therefore you need to print 5 three times,

// printSortData will output to output file,

// one data per text line.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Algorithm steps:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step 1: - open the input file

- findMinMax <-- read and find out the minimum and the maximum integer in the input file

- close the file

step 2: use constructor to dynamically allocate the BucketAry accordingly

and initialize the bucketAry to zero.

step 3: re-open the input file

step 4: data <-- read one data item at a time from input file,

index <-- getIndex (data)

increment the BucketAry [index] by 1

step 5: repeat step 4 until the input file is empty

step 6: open output file

step 7: printSortData

step 8: close input file and output file.