**Reza Shisheie**

**Homework #4**

**Due: April 27th 2017**

Question 1:

**package** homework4\_1\_test;

**import** java.util.LinkedList;

**import** java.util.Random;

**public** **class** RadixSort {

// Define the length of the array which stores integers.

**private** **static** **final** **int** ***LENGTH*** = 10;

// Define the bucket. Since it is Redix, the buckets are 10

**private** **static** LinkedList[] *q* = {

**new** LinkedList(), // 0

**new** LinkedList(), // 1

**new** LinkedList(), // 2

**new** LinkedList(), // 3

**new** LinkedList(), // 4

**new** LinkedList(), // 5

**new** LinkedList(), // 6

**new** LinkedList(), // 7

**new** LinkedList(), // 8

**new** LinkedList() // 9

};

**public** **static** **void** main(String[] args)

{

// Random List.

Object[] list = **new** Object[***LENGTH***];

// Generate a random list of numbers between 0 to 1,000,000

**for**(**int** r=0; r < ***LENGTH***; r++){

list[r] = **new** Random().nextInt(1000 \* 1000);

}

System.***out***.println("Unsorted list: ");

**for**(**int** i=0; i < list.length; i++){

System.***out***.println(list[i]);

}

// Sort the list.

Object[] sortedList = *sort*(list);

System.***out***.println(" ");

System.***out***.println("Sorted list: ");

// Print the sorted list.

**for**(**int** i=0; i < sortedList.length; i++){

System.***out***.println(sortedList[i]);

}

}

// main method to sort the integers

**public** **static** Object[] sort(Object[] list)

{

// Get the maximum number of digits in a number in the list.

**int** maxDigits = *getMaxDigits*(list);

// Iterate through the radix depending on max digits.

**for**(**int** r=1; r <= maxDigits; r++){

// Iterate through every number.

**int** radix;

**for**(**int** n=0; n < list.length; n++){

// Figure out what bucket to put it into.

radix = *getDigitAt*(Integer.*parseInt*(list[n].toString()), r);

// Put it into it's bucket accordinmaxDigits = getMaxDigits(list);g to the radix.

*q*[radix].add(list[n]);

}

// Go through the queues and put the numbers back into the list.

**int** a=0;

**for**(**int** k=0; k < *q*.length; k++){

// Go through every element in the queue.

**while**(*q*[k].peek() != **null**){

list[a++] = *q*[k].poll();

}

}

}

// Return the list, it is now sorted.

**return** list;

}

// get the maximum number of integers in a list

**public** **static** **int** getMaxDigits(Object list[])

{

// Define the max digits.

**int** maxDigits = 0;

// Iterate through the list.

**int** digits;

**for**(**int** i=0; i < list.length; i++){

// Cast the number to a string.

digits = *getDigits*(Integer.*parseInt*(list[i].toString()));

// Compare the lengths.

**if**(digits > maxDigits){

maxDigits = digits;

}

}

// Return the max digits.

**return** maxDigits;

}

// get the number of digits in an integer recursively

**public** **static** **int** getDigits(**int** i)

{

**if**(i < 10){

**return** 1;

}

**return** 1 + *getDigits*(i / 10);

}

// get the target digit in an integer

**public** **static** **int** getDigitAt(**int** number, **int** radix)

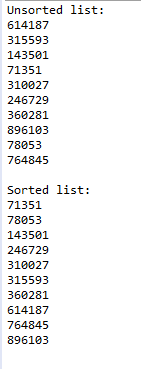
{

**return** (**int**)(number / Math.*pow*(10,radix-1)) % 10;

}

}

Question 1 Output:



Question 2:

**private** **class** MyListIterator **implements** ListIterator<E> {

**private** Node current;

// your own class is going to give this out

**public** MyListIterator(Node current) { **this**.current = current; }

// current will never be null

**public** **boolean** hasNext() { **return** current.next != **null**; }

**public** **boolean** hasPrevious() { **return** current.prev != **null**; }

**public** E next() {

**if**(!hasNext()) { **return** **null**; }

current = current.next;

**return** current.data;

}

**public** E previous() {

**if**(!hasPrevious()) { **return** **null**; }

current = current.prev;

**return** current.data;

}

}

Question 3:

To get the number of leaves, a method can be define which recursively goes to the branches of the tree and returns 1 and add them together. The result would be number of leaves

**public** **int** getNumberOfLeaves() {

**return** getNumberOfLeaves(root);

}

/\*\* Returns the number of leaf nodes \*/

**public** **int** getNumberOfLeaves(TreeNode<E> root) {

**if** (root == **null**) **return** 0;

**return** root.left == **null** && root.right == **null** ? 1 :

getNumberOfLeaves(root.left) + getNumberOfLeaves(root.right);

}