**Project 1: CS/CE/SE 3345: Data Structures and Algorithm Analysis**

**Purpose: Analysis of Sorting Techniques Due Date: Sun 9/26 at 11:59PM**

**Overview:** One of the most important ADTs is the Dictionary and one of the most studied problem is sorting. In this assignment, you will analyze multiple implementations of sorting algorithms.

Write a program to perform analysis on various sorting algorithms utilizing 4 different data types. Sorting algorithms code is provided with this assignment (insertionSort.java, selectionSrot.java, quicksort.java, HeapSort.java, RadixSort.java, MergeSort.java)

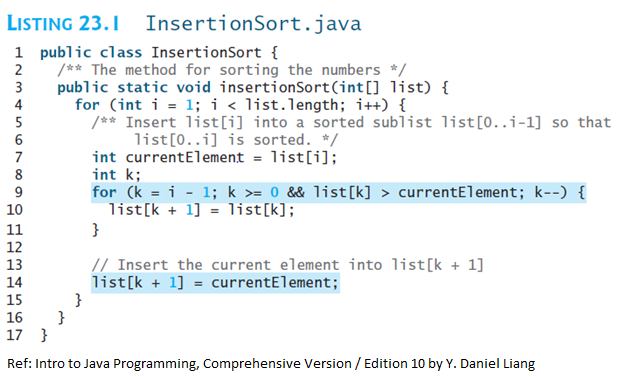
Create and submit a report discussing the analysis at each iteration. Clearly define your approach, challenges, and assessment. Total Time is (Start – End) in micro or milli seconds.

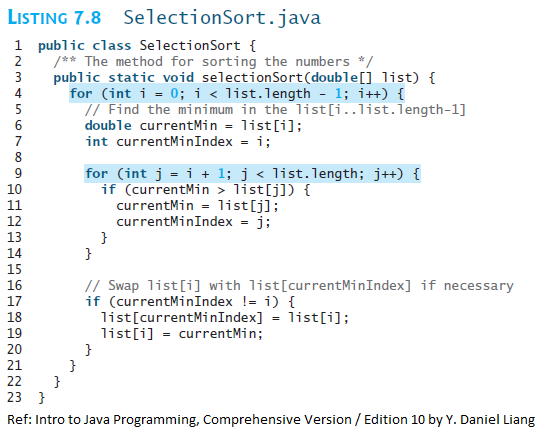
Students posts / questions:

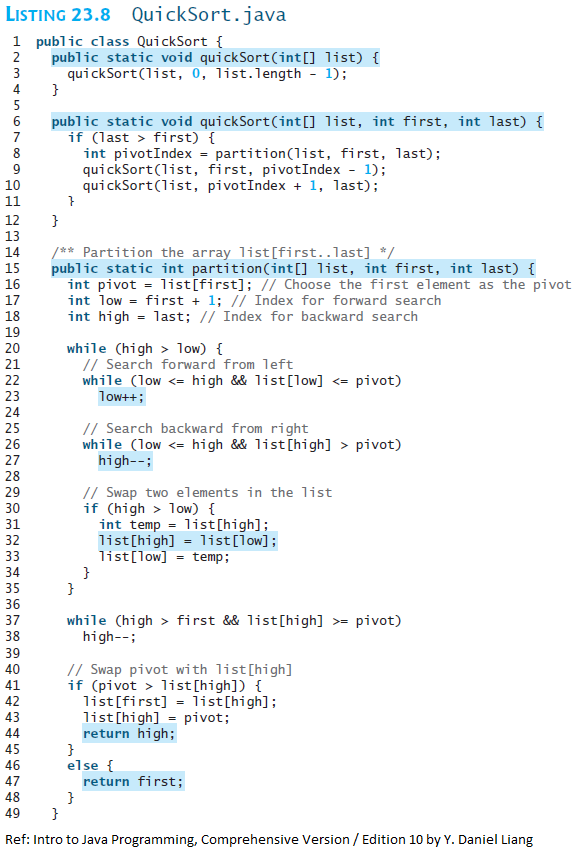
**Quick Sort**:

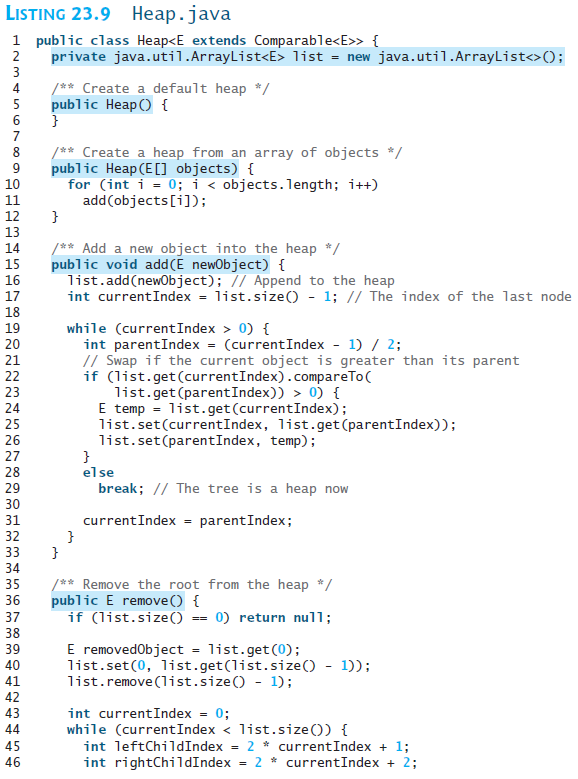
1. **Student 1**: The quick sort that Dr. Khan has provided will run into an infinite loop around 14k elements. So can we just rewrite the quick sort?
2. **Student 2**: Did you get Stack Overflow? If you did, then it's because the default 1mb of memory storage isn't enough to process more than 14k elements. I had the same problem, and I fixed it by changing the memory storage in my IDE to 3mb under VM argument using the command -Xss3m.
3. **Student 3**: it only has issues during In Order and Reverse Order. I would imagine this is because the pivot selection for this implementation is resulting in there being one side that is very small and one that is very large when the list is sorted.
4. **Student 4:** I changed the pivot to (front + back)/2 instead of front. This is the best for our project in my opinion because it is easy to calculate and solves the issues we are having.
5. **Student 5**: Are you sure it's the sorting algorithm? Me and my buddy found that it was the limits of our printing functions preventing us from checking sort. Once I switched to making my check a for loop of println's, it was able to sort and print at least a billion elements.

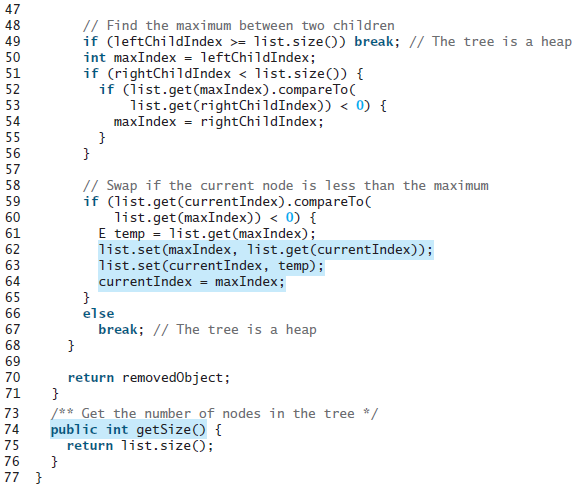
**Primitive vs Generic Type:** **Box primitive int to Integer object for algorithms requiring generic**

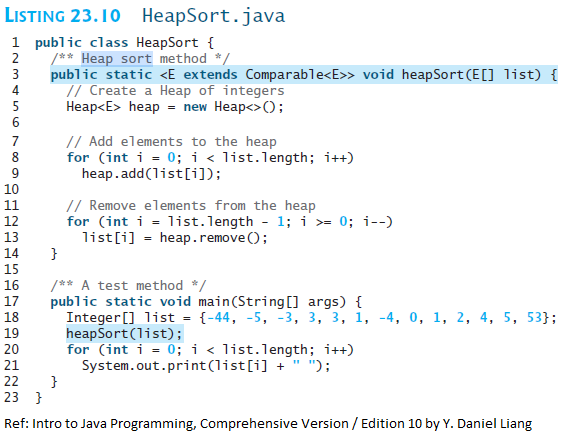


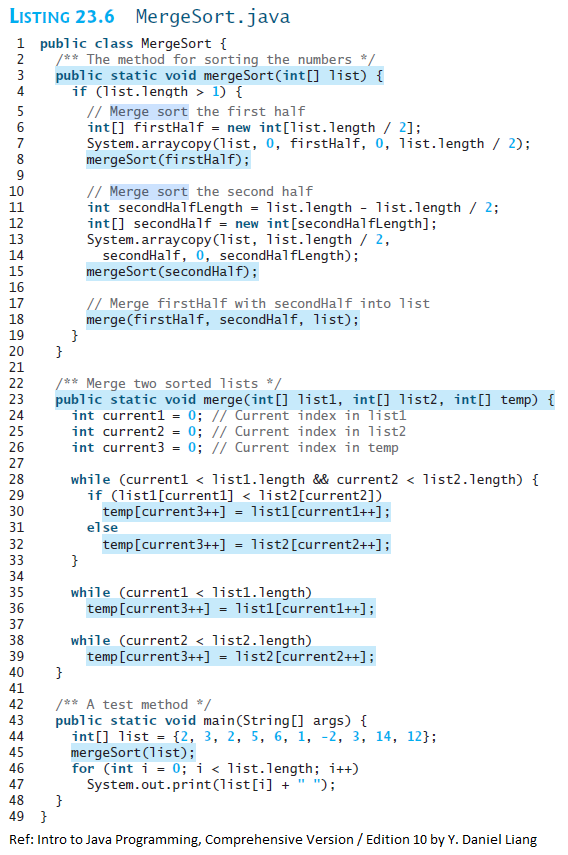


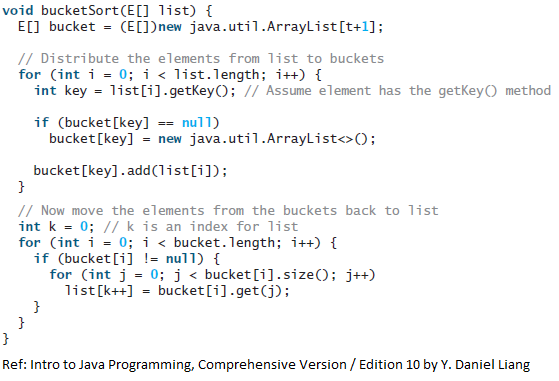












/\*\*\*\*\*\*\*\*\*\*\*\* RADIX SORT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.io.\*;

import java.util.\*;

class Radix {

static int getMax(int arr[], int n){

int mx = arr[0];

for (int i = 1; i < n; i++)

if (arr[i] > mx)

mx = arr[i];

return mx;

}

static void countSort(int arr[], int n, int exp) {

int output[] = new int[n];

int i;

int count[] = new int[10];

Arrays.fill(count,0);

for (i = 0; i < n; i++)

count[ (arr[i]/exp)%10 ]++;

// Change count[i] so that it contains actual position of this digit in output[]

for (i = 1; i < 10; i++)

count[i] += count[i - 1]; // Build the output array

for (i = n - 1; i >= 0; i--){

output[count[ (arr[i]/exp)%10 ] - 1] = arr[i];

count[ (arr[i]/exp)%10 ]--;

}

for (i = 0; i < n; i++)

arr[i] = output[i];

}

static void radixsort(int arr[], int n)

{ // Find the maximum number to know number of digits

int m = getMax(arr, n);

for (int exp = 1; m/exp > 0; exp \*= 10)

countSort(arr, n, exp);

}

static void print(int arr[], int n) {

for (int i=0; i<n; i++)

System.out.print(arr[i]+" ");

}

public static void main (String[] args) {

int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};

int n = arr.length;

radixsort(arr, n);

print(arr, n);

}

}