CSCE 222 [503] Discrete Structures for Computing
Spring 2015 – Philip C. Ritchey

Extra Credit Assignment: Algorithms

Due date: Electronic submission of this extra credit assignment is due on **3/15/2015 (Sunday) before 11:59 p.m.** on `http://ecampus.tamu.edu`. Submit a signed copy of this sheet to me by 3/15/2015 (Sunday) before 11:59 p.m..

Name: (type your name here)

Resources. Discrete Mathematics and Its Applications by Rosen, (additional people, books, articles, web pages, etc. that have been consulted when producing this assignment. Failure to cite sources will result in failure to pass this class.)

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this assignment. I understand that failure to cite my sources will result in failure to pass this class.

Signature: _____

Problem 1. (5 pts each) Write a program (using C, C++, Java, or Python) that reads a list of integers from standard input (numbers are separated by newlines and the list is terminated by the ASCII EOT character), sorts the list using any of the following algorithms, and outputs the sorted list to standard output (numbers separated by newlines). The name of your program should be the same as the name of the sorting algorithm.

1. Bubble Sort (named `BubbleSort`)
2. Insertion Sort (named `InsertionSort`)
3. Selection Sort (named `SelectionSort`)
4. Binary Insertion Sort (named `BinaryInsertionSort`)
5. Radix Sort (named `RadixSort`)
6. Quick Sort (named `QuickSort`)
7. Merge Sort (named `MergeSort`)

Problem 2. (10 pts each) Add to your programs the ability to accept a command line argument `-time` that causes the program to output the amount of time spent sorting (including the time spent reading the input) instead of the sorted list. Using lists of random integers of various lengths (ranging from 1 element to 10^5 elements; write a program to generate these lists for you), collect timing data about how long it takes each sorting algorithm to sort a list of each various size. Plot the results (using whatever software you want, e.g. Excel or Matlab or Python or Gnuplot): put input size on the x-axis and put time on the y-axis. Plot all the results on a single graph and label (or use a legend) each line with the name of the sorting algorithm and a big-O estimate for the runtime as a function of the size of the input.