CS 251, Fall 2007
Voicu Popescu
Assignment 9
Due Friday November 9, 11:59PM

Sorting

1. Implement **void quicksort**(unsigned int *_a_, int _n_) as a "C" style function in the file quicksort.cc
    a. The function sorts the array _a_ of _n_ positive integers.
    b. The main function in quicksort.cc reads from the file "quicksortinput" the _n_ integers and writes the _n_ integers to the file "quicksortoutput" in ascending sorted order. See quicksortsampleinput and quicksortsampleoutput for formatting details. Please make sure that your program can EXACTLY reproduce that output file given the input file. You may want to use the `diff' program to test this.
    c. The pivot should always be chosen as the first element of the array.
    d. Run your function on 10 pseudo-randomly generated arrays of integers with lengths 1,000, 2,000, …, and 10,000 (i.e. one thousand through ten thousand). **Count the actual number of array element comparisons and graph it as a function of _n_. Measure the actual running time and graph it as a function of _n_.** Titles, axes labels, units, etc are required. Place both graphs in the file "quicksort.pdf"
    e. See the given quicksort.cc file for how to determine running times. You can modify this file as you wish but I should be able to run it to get your output in the file specified.

2. Implement **void radixsort**(unsigned int *_a_, int _n_) as a "C" style function in the file radixsort.cc
    a. The function sorts the array _a_ of _n_ positive integers.
    b. The elements of the array are positive and are represented with 32 bits.
    c. The main function in radixsort.cc reads from the file "radixsortinput" the _n_ integers and writes the _n_ integers to the file "radixsortoutput" in ascending sorted order. See radixsortsampleinput and quicksortsampleoutput for formatting details. Please make sure that your program can EXACTLY reproduce that output file given the input file. You may want to use the `diff' program to test this.
    d. Run your function on 10 pseudo-randomly generated arrays of integers with lengths 1,000, 2,000, …, and 10,000 (i.e. one thousand through ten thousand). **Count the actual number of array element to bucket assignments and plot it in a graph as a function of _n_. Measure the actual running time and graph it as a function of _n_.** Titles, axes labels, units, etc are required. Place both graphs in the file "radixsort.pdf"
    e. See the given radixsort.cc file for how to determine running times. You can modify this file as you wish but I should be able to run it to get your output in the file specified.

3. Turn in instructions:
    a. Assignment specific:
        i. Turn in files: quicksort.cc, quicksort.pdf, radixsort.cc, radixsort.pdf
        ii. A Makefile is not required.

iii. DO NOT change any of the function signatures. Doing so will most likely lead to a compile error and a 0 for your overall program grade.
iv. Do not partition your code into files we haven't asked for.
v. Your code is REQUIRED to compile with /opt/csw/gcc3/bin/g++ (gcc version 3.4.5) on the borg machines. If your code doesn't compile with this compiler, then even if it compiles with another, your code will still be considered noncompiling.
vi. Remember that assignment-specific turnin instructions override general instructions.
vii. Directory structure: all files in the directory lab9
viii. turnin command executed in the directory containing your lab9 directory:
- turnin -c cs251 -p lab9 lab9

b. General:
See instruction at http://www.cs.purdue.edu/cgvlab/courses/251/As/turnin.html
Make sure to check it EVERY TIME before submitting a new assignment. It is updated periodically.