Homework #2: Sorting Algorithms (100 pts)

For this assignment, you must download the provided **C++ source code** (and **Makefile**) and *implement* **three** of the *sorting* algorithms covered in class:

1. **Insertion Sort**
2. **Quick Sort**, and
3. **Radix Sort**

In particular, you must implement these algorithms *within* the provided **C++ source code** and using the given **Makefile**. Your implementation of **Insertion Sort** must go in ***insertionSort.cpp*,** your solution for **Quick Sort** must go in ***quickSort.cpp***, and your implementation of **Radix Sort** must go in ***radixSort.cpp***.

NOTES:

- You are **NOT** allowed to modify the provided **main.cpp**, **sort.h**, or **Makefile**.
  - I will test your code using "fresh" (original) copies of these files.
  - For example, if you submit a **main.cpp** file, I will overwrite it before testing.

- **main.cpp** is a *driver* program that takes **N** integers on *stdin*, **sorts** the numbers using all **3 sorting algorithms**, then outputs the algorithm's **runtime** (in microseconds) and **correctness** (i.e., if input has been correctly sorted).

- Obviously, your solutions **CANNOT** use C++'s built-in **sort( )** function.

- The provided code will **compile** and **execute** "out of the box".
  - But will not sort anything… (that's your job!)

```
//use Makefile to compile code (ignore output by redirecting to /dev/null)
UNIX> make > /dev/null


UNIX> ./main
8 6 7 5 3 0 9<ENTER><CTRL-D>
   algorithm            N       runtime         test
=====================================================
   insertion            7          0.00     NOT SORTED
       quick            7          0.00     NOT SORTED
       radix            7          0.00     NOT SORTED
```

<u>EXAMPLES</u> (with all sorting algorithms correctly implemented)

```
//use "clean" target of Makefile to delete all .o files and main executable
UNIX> make clean
rm *.o
rm main


//compile everything using the provided Makefile
UNIX> make
g++ -c main.cpp
g++ -c insertionSort.cpp
g++ -c quickSort.cpp
g++ -c radixSort.cpp
g++ -o main main.o insertionSort.o quickSort.o radixSort.o


//run "main" executable created using the provided Makefile
UNIX> ./main
8 6 7 5 3 0 9<ENTER><CTRL-D>
   algorithm            N      runtime         test
======================================================
   insertion            7         0.00         SORTED
       quick            7        16.00         SORTED
       radix            7        51.00         SORTED


//use 'seq' command to generate numbers 1 through 5
UNIX> seq 1 5
1
2
3
4
5


//"pipe" the output of 'seq' command to 'gshuf' to shuffle randomly
UNIX> seq 1 5 | gshuf
2
4
3
5
1


// use 'seq' to create numbers 1 through 5, pipe it to gshuf
//    then pipe output of gshuf to main executable
UNIX> seq 1 5 | gshuf | ./main
   algorithm            N      runtime         test
======================================================
   insertion            5         1.00         SORTED
       quick            5         1.00         SORTED
       radix            5        24.00         SORTED
```

```
//generate 1000 random numbers and pipe to main executable
UNIX> seq 1 1000 | gshuf | ./main
   algorithm           N      runtime         test
=====================================================
   insertion        1000      1473.00         SORTED
       quick        1000       183.00         SORTED
       radix        1000       663.00         SORTED
```

```
//generate 10000 random numbers and pipe to main executable
UNIX> seq 1 10000 | gshuf | ./main
   algorithm           N      runtime         test
=====================================================
   insertion       10000    151988.00         SORTED
       quick       10000      2369.00         SORTED
       radix       10000      4717.00         SORTED
```

```
//generate 10000 sorted numbers and pipe to main executable
//          notice how quicksort suffers..
UNIX> seq 1 10000 | ./main
   algorithm           N      runtime         test
=====================================================
   insertion       10000        91.00         SORTED
       quick       10000    140202.00         SORTED
       radix       10000      6194.00         SORTED
```