Algorithms Homework Assignment #1

"Measure and compare running time of Merge Sort, Insertion Sort, and Selection Sort" (3 points)

Due date: Wednesday, March 21st 2018, 9AM.
- Submit hardcopy during class and softcopy on the server.

In this homework assignment, you'll implement the "Merge Sort", "Insertion Sort", and "Selection Sort" algorithms, measure their running time, and plot a graph that compares the running time of the three algorithms.

Here are the steps and requirements on what you'll need to do.

Download the sample text file "hw1_input.txt". This file contains '**K**' integers in random order. You should use this file as input to test your program. However, your program should work with other input files with different K and different order of integers also.

Your task is to

1. Sort the first 'N' numbers in a file, in descending order, using "Merge Sort", "Insertion Sort", "Selection Sort" algorithms,
2. Measure their running time for N = 10000 ~ 1,000,000 and
3. Plot a graph that compares the running time of the three algorithms.

You must implement using **C**, and you must implement three separate programs, "merge_sort.c", "insertion_sort.c", and "selection_sort.c". The file names must be exactly as given, otherwise it won't be graded and you will get zero points. Your program should compile with gcc compiler.

Your program must take two command line arguments: <N> and <input filename>

- Ex> `./<your_program> <N> <input_file>`
- Ex> `./merge_sort  10000 hw1_input.txt`
- Ex> `./insertion_sort  10000 hw1_input.txt`
- Ex> `./selection_sort  100000 hw1_input.txt`

Your program should do the following

- For a given N value, read the first N integers from the input file, put them into an array of integers, and sort them using each sorting algorithm.
  - if N > K, then your program should sort all K numbers in the file correctly.
- Your program must output the sorted result as well as the running time of the program in milliseconds. Make sure that your sorted result is correct for N = 1000000 using the sample input file.

```
$ ./insertion_sort  1000000 hw1_input.txt
1000000
999999
…
2
1
Running time = xxxxxx ms
$
```

To measure running time,

- You may use the 'clock()' function or the 'gettimeofday()' function in your C program.
- Please be careful about measuring time for small N values; the time will be very small, you need to think carefully about how to measure that small amount of time accurately.

Run your experiment with at least N = 10000, 100000, 200000, 500000, <student ID>%1000000, 1000000, (possibly more) and measure the running time for both "Merge Sort", "Insertion Sort", and "Selection Sort". You may run experiments with more N values to get a smoother plot.

Plot two graphs that compares the running time of the three sorting algorithms.

- With N on the x-axis, running time on the y-axis (in milliseconds).
- With log(N) on the x-axis, log(running time) on the y-axis
- Each graph must plot both "Merge Sort", "Insertion Sort", and "Selection Sort".
- N values must include "N = 10000, 100000, 200000, 500000, <student ID>%1000000, 1000000", and you may use more N values for smoother and better looking plot.
- Note that x-axis values are not equally spaced. Make sure you have correct scale on the x-axis!

You program must work with other input files as well, not just 'hw1_input.txt'. We will test and grade your program with other input files with different number of integers. Also, your program must gracefully handle all exception/error cases such as N > K, N < K, no input file, incorrect command line arguments, etc.

What and how to submit

- You must submit both a hardcopy report and softcopies of all the files;
- Hardcopy report should include the graphs that compare the running time of the three sorting algorithms, and your conclusion. Hardcopy report should be submitted during class.
- Here is the instruction on how to submit the softcopy files :
  ① Login to your server account at nsl2.cau.ac.kr.
  ② In your home directory, create a directory "submit_alg/submit_<student ID>_hw1" (ex> "/student/20149999/submit_alg/submit_20149999_hw1")
  ③ Put your "merge_sort.c", "insertion_sort.c", and "selection_sort.c" file in that directory.
- Your submission will not be graded if you do not follow the instructions exactly.

Other requirements:

- Your program must run on our class Linux server at nsl2.cau.ac.kr.
- Your code must include your name and student ID at the beginning of the code as a comment.
- Your code should be easily readable and include sufficient comments for easy understanding.
- Your code must be properly indented. Bad indentation/formatting will result in score deduction.
- Your code should not include any Korean characters.
- Your program should work regardless of whether the input file format is Windows/Linux/MacOS based.
  - ✓ That is, input file may have Windows or Linux or MacOS line ending format (\n, \r\n, \r).

Grading criteria:

- You get 3 points
  - ✓ if all your programs work as requested, AND if you meet all the requirements, AND
  - ✓ if your hardcopy report is well written.
- Otherwise, partial deduction may apply.
- You may get optional extra credit of up to 1 point if you do the optional extra credit task.
- No delayed submissions are accepted.
- Copying other student's work will result in negative points.
- Code that does not compile or code that does not run will result in negative points.

[Optional] Extra credit task:

- Do the same thing as above also in **Java or Python** (in addition to C).
  - ✓ You can select either Java or Python. (I prefer Python, but selection is up to you)
    ① For Java, three files should be 'merge_sort.java', 'insertion_sort.java', and 'selection_sort.java'.
    ② For Python, three files should be 'merge_sort.py', 'insertion_sort.py', and 'selection_sort.py'.
- Plot graphs that compare not only the three sorting algorithms, but also the performance differences between C and {Java or Python}. Discuss the result.
- Include everything in your submissions.