EECS 233 Homework 1

General requirements:

- Due at 11:00 PM on the posted due date.
- Include your name and network ID as a comment at the top of all of your programs.
- Create a typed document (.docx or .pdf) for the report with your name and network ID as a comment at the top.
- Upload your document and all .java files as a .zip file to Blackboard. Do not use other formats such as .rar.
- All work should be your own, as explained in the Academic Integrity policy from the syllabus.

Instructions: Revise the three programs provided (Example1.java, Example2.java, and Example3.java) to automatically (using a loop) print the run times for at least 6 equally spaced *N* values. You may keep the programs separate or combine them into a single program (your choice). Below is an example of how your output might appear for one of the algorithms:

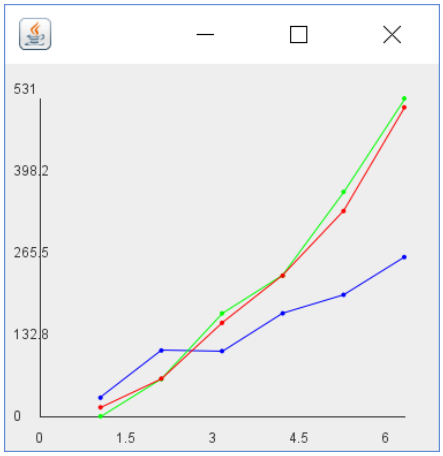
```
------- Algorithm #1 --------
N = 10000000, time = 31 msec
N = 20000000, time = 110 msec
N = 30000000, time = 109 msec
N = 40000000, time = 172 msec
N = 50000000, time = 203 msec
N = 60000000, time = 266 msec
```

Type a report as described below:

1. Copy/paste the output of the program(s), showing the specific algorithm (1, 2, or 3), each *N* value, and each run time. Create output for the following two conditions:
   a. Choose *N* values such that all of the run times are less than 1 second, as shown in the example above.
   b. Choose *N* values such that all of the run times are more than 1 second.

   Notes:
   - You will have a total of 6 blocks of output (2 for each algorithm). For convenience, you may run the program once for condition (a) and then manually change the *N* values before running the program again for condition (b).
   - The *N* values do not need to be the same for all of the algorithms. For example, you may use larger *N* values for faster algorithms and smaller *N* values for slower algorithms in order to have run times that are convenient for comparison.

2. Using any software you choose, create one graph of the run times for each of the two conditions above (2 graphs total). Each graph should have one curve for each algorithm. To visually compare the algorithms, use indexes (1, 2, ...6) for the x-axis values instead of actual *N* values, since the *N* values may be different (see note above). Provide some sort of legend (typed is OK) indicating which curve corresponds to each algorithm. EXTRA CREDIT (10 points): Use the program Grapher.java that is provided to automatically create the graphs. This is not required. On the right is an example (note that no legend is given here, but you should provide a legend in your report). Disclaimer: yes, we know it's not perfect, but it's better that the non-existent library.



3. Describe how the three algorithms compare qualitatively, and whether the results agree or disagree with the expected Big-O run time. Use at least 3 complete sentences.

4. Describe any qualitative differences between the two conditions in (1). Is there any advantage to using run times greater than 1 second? Use at least 2 – 3 complete sentences. Any reasonable description will receive full credit. Though not required, you may offer an explanation for any differences you see.

Grading Rubric:

Item	**Points**
Coding: Proper use of sample programs (~7 points per algorithm)	20
Coding: Creating N values with loop (~7 points per algorithm)	20
Coding: Printing reasonable output (~7 points per algorithm)	20
Report item #1	10
Report item #2	10
Report item #3	10
Report item #4	10
Total	100

Extra credit: +10 points for successful use of Grapher.java.