Run Time Analysis

First off, we want to write a program `MyRand` which will create two data sets of random integers of sizes 10,000 and 40,000. Display this program by typing `cat myRand.java`

Notice the `for` loop that will run *size* times where *size* is an integer typed at the command line upon execution. In the body of that loop, be sure you understand the one line of code which will print a random integer between 1 and 500 to the screen for each loop iteration. Our sequence is inclusive (i.e., includes the 500).

Compile this program, and run it using the command :

```
java MyRand 10000 > data1
```

This instructs the operating system to send all output to the file `data1` rather than the screen. Check the file `data1` and make sure it contains 10,000 random integers in your desired range.

Now, re-run your program using a command that will give us a file of 40,000 integers :

```
java MyRand 40000 > data2
```

Now, let's determine the actual amount of system run-time that it takes to sort the data from both of our files using the bubble sort algorithm (an $O(n^2)$ algorithm). To do this, we need to use the *Linux* `time` command. This command will return the time in seconds that it takes the CPU to complete your program.

The program `Bubble` is an executable program which reads in integers from the keyboard, and sorts them using the bubble sort algorithm. This program is set up to take the desired array size you plan on sorting from the command line. So, here is the command we need to type to calculate the run time of a bubble sort on an array of size 10,000 is :

```
time java Bubble 10000 < data1
```

The total time it takes this to run will be the first float value on the first line prior to ``user." It should be around two seconds to sort 10,000 integers using a bubble sort.

Run-Time Analysis was based on our in-class Run-Time Notation worksheet. Once we set how as you increase the amount of input n, you increase the run time.
The Run-Time Analysis helped you see this in action by looking at the exact time it took us to run a piece of software.