Exercise 01

This is our first exercise. It's being given out on the first day of class, so it's called exercise 01. We'll keep the exercise numbers aligned with the lecture numbers. So, if we don't have an exercise for a particular day, the exercise numbering will just skip that day.

For this exercise, you get to fill in the missing parts of a simple C program. This will give you a chance to type in some C syntax yourself and to see some of the areas where the Java you already know will translate directly to C.

The program you're working on is called arithmetic.c; its job is to add up all the numbers from 1 to 10^9 (sum of an arithmetic series from one to one billion). You'll find a starting point for this program on the course homepage in Moodle, along with a copy of an expected output file. If you download these files from the Moodle page, be sure you don't copy-and-paste the contents of the expected output file. Download by right-clicking and choosing to save a copy of the file instead (this might vary a bit from browser to browser). If you copy-and-paste the contents of this file, you might accidentally change something in it (like you might miss the newline at the end of the line of output).

Or, if you're already logged in on one of the common platform systems, you can enter the following curl commands to download copies of the two files you need and put them in your current directory.

```
curl –O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise01/arithmetic.c
curl –O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise01/expected.txt
```

In the source code of arithmetic.c, I've given you comments describing the code you need to add. In general, I'll mark places where you need to add code with a comment that looks like `// ...` or like `/* ... */`

Once you've completed the program, you can build and run it with commands like the following. It may take a couple of seconds to complete execution, but it should print a line like the one below if it's working correctly:

This is the shell prompt.   Here, we're compiling the program.

```
eos$ gcc –Wall –std=c99 arithmetic.c –o arithmetic
eos$ ./arithmetic
500000000500000000
```

Here, we're running it.

Here's its output.

Keep in mind that this exercise will be graded automatically using a grading script. To get full credit, you'll need to make sure your source file is named exactly the right thing (including capitalization), that it compiles cleanly, prints exactly the right output when run and exits with successful exit status. You probably don't need it for this exercise, but I'm providing a sample output file, expected.txt, showing exactly what output your program is expected to print. If you put this file in the same directory as your program, you should be able to run commands like the following to check your program's exit status and make sure it prints exactly the right output:

Running the program with output saved to a file. (the >| says overwrite the output file)

```
eos$ ./arithmetic >| actual.txt
eos$ echo $?
0
eos$ diff expected.txt actual.txt
eos$
```

Checking the exit status. Good, it's zero. (have to do this right after you run)

Comparing actual output with expected. Good. No differences reported.

Once you have completed your program and you're happy with how it runs, submit your arithmetic.c source file to the exercise_01 assignment in Moodle. Be sure to submit before 11:00 pm on Sunday.