CSCI 4060U – Laboratory #1
Introduction to OpenMP Programming in C
Lab Due: <u>Friday, January 24, 2025 at 11:00pm (Canvas)</u>

Introduction

The main purpose of this lab is to introduce you to writing, compiling and running OpenMP programs in C.

Activity #1

In the first activity, we will ensure that you can compile and execute the `helloworld.c`[1] program from Lecture 3. To compile the program, run `gcc` as follows:

```
command$ gcc –fopenmp helloworld.c
```

If you are using Mac OSX you should use:

```
command$ gcc –Xpreprocessor –fopenmp –lomp helloworld.c
```

Next, run the program and observe the output. You will also need to show you teaching assistant that your program runs correctly.

Activity #2

In the second activity, you will parallelize the program `pi_serial.c`[2]. This program approximates π as a sum of rectangles. Your goal is to parallelize the algorithm to use **_two_** threads for calculating the rectangle heights. You will need to use the following compiler directive to indicate the parallel code block:

```
#pragma omp parallel
{
  ...
}
```

You will also need the following OpenMP functions:

- `omp_set_num_threads(int x)` – sets the number of threads
- `int omp_get_num_threads()` – returns the number of threads
- `int omp_get_thread_num()` – returns the current thread ID

---

[1] available at: https://www.sqrlab.ca/courses/csci4060u-w25/exercises/helloworld.c
[2] available at: https://www.sqrlab.ca/courses/csci4060u-w25/exercises/pi_serial.c

Once you have parallelized the program, you will need to assess its performance. Using the `time` command run the `pi_serial.c` and `pi_parallel.c` executables to obtain the real, user and system times of each. These times need to be included as a comment in `pi_parallel.c`.

```
command$ time ./a.out
```

Submission. You should submit your *source file* for Activity #2 (`pi_parallel.c`) through the lab drop box in Canvas.