# CSC/ECE 506: Architecture of Parallel Computers
# Program 2: OpenMP programming
# **Due: Friday, June 13, 2025**

This set of six short programming exercises will familiarize you with OpenMP and the shared-memory programming model.

**Procedure**

1. Log into login.hpc.ncsu.edu with your unity id/password via ssh (e.g., via PuTTY on Windows)
2. Download the OpenMP example tar-ball by executing
   `wget https://www.csc2.ncsu.edu/faculty/efg/courses/506/sum25/www/homework/p2/program2.tgz`
3. Use this command to decompress the tar-ball: `tar zfvx program2.tgz`
4. `cd` into `program2`, and then into each of its subdirectories in succession.
5. Check that you can compile each example by executing `make` in each subdirectory.

**Deliverables**

Compress the program2 directory into a tgz file using the following command:

```
tar -xvf program2.tgz
```

and submit it to Moodle.

Your .tgz file should contain—

1. The files specified in square brackets next to each question, with their requested contents.
2. A **`report.pdf`** file that includes all the explanations that you are asked to provide below.

To avoid breaking the grading scripts and causing extra hassle for the TA, please follow these naming conventions strictly.

**Problem 1.** *(5 points each)* (a) Uncomment the pragma in line 9 of **`p1.c`**, recompile it, and check its output. How many "part2"s do you get with or without the pragma? Why?

(b) Modify the pragma in line 9 of **`p1.c`** to output three "part2"s, and write the modified file to **`p1b.c`**.

(c) Uncomment line 6 and line 12 of **`p1.c`** and add one OMP directive of the form `omp_set_num_threads()` in your solution of part (b) to output two "part1"s, three "part2"s, and two "part3"s. Put your code into **`p1c.c`**. Explain your solution.

**Problem 2.** *(10 points)* (a) Modify line 7 and fix line 9 and line 10 of **`p2.c`** to generate the following output, except that it is OK if your thread IDs print in a different order: [Put your code into **`p2a.c`**]

```
Hello from thread 0 of 10
Hello from thread 1 of 10
Hello from thread 2 of 10
Hello from thread 3 of 10
Hello from thread 9 of 10
Hello from thread 4 of 10
Hello from thread 5 of 10
Hello from thread 6 of 10
Hello from thread 7 of 10
Hello from thread 8 of 10
```

(b) *(5 points)* Modify the above program to use the default number of threads. [**`p2b.c`**]

**Problem 3.** *(10 points)* Fix line 13 of **p3.c** to set all entries in the array to "1" with the "private" clause. Explain why the change is needed. [**p3.c**]

**Problem 4.** *(5 points each)* (a) Explain why p4**.c** does not calculate the correct "parallel sum" and fix it using the `reduction` clause. [**p4a.c**]

(b) Fix p4.c using the `atomic` clause (instead of fixing it the way you did in part (a)). [**p4b.c**]

(c) Now fix p4.c by using the `critical` clause (instead of the fixes of parts (a) and (b)). [**p4c.c**]

(d) Explain the difference in part (b) and part (c).

**Problem 5.** *(10 points)* Fix line 6 and 8 of **p5.c** to use `private`, `lastprivate`, `firstprivate` clauses one after the other. Copy the outputs in **report.txt** file for each clause and explain the differences if any. [**p5.c**]

*Note: There should be one p5.c file with instructions in the comment on how to use private, lastprivate and firstprivate.*

**Problem 6.** (a) *(20 points)* Calculate π using this Gregory-Leibniz series:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \frac{\pi}{4}$$

Start from **p6.c**. Use "`omp parallel for`" to parallelize the **for**-loop with some other OMP directives to get the correct estimation. In addition, measure the performance improvement of OpenMP by inserting calls to `omp_get_wtime()` in your program. [**p6a.c**]

(b) *(10 points)* Use static and dynamic scheduling mechanism for the changes done in part (a). Compare the timing statistics and explain why one mechanism performs better than the other. [**p6b.c**]