Before Starting Homework

- Do not upload to generative AI.
- Complete the Programming Assignment before beginning the written portion of the homework.
- Answer all questions below.
- **Type** answers and upload to Canvas in PDF format.

Question 1 : 1 : Scalability in Calculating PI

For the programming assignment portion of this homework, you are approximating PI on a number of threads. To guarantee correctness, you are implementing a set of different locks. An example file is provided for you in `examples/example.cpp`. This example calls your function to calculate PI with a variety of different thread counts. The total number of samples is constant, so as thread counts increase, the number of samples per thread decreases.

The example is compiled with each of your lock implementations:

- `build/examples/example`: linked with the pthread mutex lock implementation as provided for you in the submodule file `mutex_lock.cpp`.
- `build/examples/example_spin`: linked with your ticket spin lock
- `build/examples/example_yield`: linked with your ticket yield lock
- `build/examples/example_queue` : linked with your queue lock
- `build/examples/example_semaphore` : linked with your semaphore lock

Compare the performance of each of the locks above by answering the following questions :

1. How many cores are available on the computer on which you are running?
2. Which of the locks is most scalable (aka best performance increases with thread count)?
3. For each lock:
   (a) What is the strong scaling limit (aka the number of threads that gets best performance)? You may need to increase the for loop within the example to test additional thread counts depending on the computer you are testing on.
   (b) Is there a point before the strong scaling limit where the scalability is greatly reduced? If so, what is this limit, and is it related to the hardware of your computer?
4. Include a graph comparing all of these timings (ideally, include one line per lock for all thread counts, all within a single figure).