Exercise 1

- Create a program that runs N threads similar to one explained in the tutorial. Each thread should output its number and some text, then exit
- Main program should inform about thread creation
- Run the program and discuss the results
- Fix the program to force the order to be strictly thread 1 created, thread 1 prints message, thread 1 exits and so on
- Hint: use gcc -pthread ex1.c to compile

Exercise 2

- Write a shell script that produces a file of sequential numbers by reading the last number in the file, adding 1 to it, and then appending it to the file. Run one instance of the script in the background and one in the foreground, each accessing the same file
- How long does it take before a race condition manifests itself? What is the critical region?
- Modify the script to prevent the race
- Hint: use ln file file.lock to lock the data file

Exercise 3 (1/2)

- Write a producer-consumer problem that uses threads and shares a common buffer. However, do not use semaphores or any other synchronization primitives to guard the shared data structures. Just let each thread access them when it wants to. Use sleep and wakeup to handle the full and empty conditions. See how long it takes for a fatal race condition to occur. For example, you might have the producer print a number once in a while. Do not print more than one number every minute because the I/O could affect the race conditions

Exercise 3 (2/2)

- Hint: you can use use global variables to implement sleep and wakeup routines
- Normally, it is done by using condition variables (https://computing.llnl.gov/tutorials/pthreads/#ConditionVariables)

Exercise 4 (Optional)

- Write a multi-threaded solution for computing the least common multiple (also called the lowest common multiple or smallest common multiple) of a series of positive integers
- Your program will take two arguments from the command line. The first argument is the name of the file that contains the positive integer numbers, and the second argument is the number of threads T to be created. You may assume that the number of threads is always smaller than the number of integers given in the input file

Exercise 5 (Optional)

- Implement dining philosophers problem