Exercise 2

1. Mutexes

1. Review, compile and run the dotprod_serial.c program. As its name implies, it is serial - no threads are created.
2. Now review, compile and run the dotprod_mutex.c program. This version of the dotprod program uses threads and requires a mutex to protect the global sum as each thread updates it with their partial sums.
3. Execute the dotprod_mutex program several times and notice that the order in which threads update the global sum varies.
4. Review, compile and run the bug6.c program.
5. Run it several times and notice what the global sum is each time? See if you can figure out why and fix it. The explanation is provided in the bug examples table above, and an example solution is provided by the bug6fix.c program.
6. The arrayloops.c program is another example of using a mutex to protect updates to a global sum. Feel free to review, compile and run this example code as well.

2. Condition Variables

1. Review, compile and run the condvar.c program. This example is essentially the same as the shown in the tutorial. Observe the output of the three threads.
2. Now, review, compile and run the bug1.c program. Observe the output of the five threads. What happens? See if you can determine why and fix the problem. The explanation is provided in the bug examples table above, and an example solution is provided by the bug1fix.c program.
3. The bug4.c program is yet another example of what can go wrong when using condition variables. Review, compile (for gcc include the -lm flag) and run the code. Observe the output and then see if you can fix the problem. The explanation is provided in the bug examples table above, and an example solution is provided by the bug4fix.c program.

3. Hybrid MPI with Pthreads

1. Your pthreads directory should contain the following 5 codes:

- mpithreads_serial.c
- mpithreads_threads.c
- mpithreads_mpi.c
- mpithreads_both.c
- mpithreads.makefile

These codes implement a dot product calculation and are designed to show the progression of developing a hybrid MPI / Pthreads program from a a serial code. The problem size increases as the examples go from serial, to threads/mpi to mpi with threads.

Suggestion: simply making and running this series of codes is rather unremarkable. Using the available lab time to understand what is actually happening is the intent. The instructor is available for your questions.

1. Review each of the codes. The order of the listing above shows the “progression”.
2. Use the provided makefile to compile all of the codes at once. The makefile uses the gcc compiler - feel free to modify it and use a different compiler.

   ```
   make -f mpithreads.makefile
   ```

3. Run each of the codes and observe their output:

Execution command	Description
mpithreads_serial	Serial version - no threads or MPI
mpithreads_threads	Threads only version of the code using 8 threads
srun -n8 -ppReserved mpithreads_mpi	MPI only version with 8 tasks running on a single node in the special workshop pool
srun -N4 -ppReserved mpithreads_both	MPI with threads using 4 tasks running on 4 different nodes, each of which spawns 8 threads, running in special workshop pool