3.21 and 3.22

I attached the code in a zip file.

4.7 A multithreaded solution provides better performance when many tasks can be performed in parallel instead of sequentially.

4.8 Heap memory and global variables are shared in a multithreaded process.

4.11 Yes, because concurrency means multiple tasks can be supported and managed, whereas parallelism means multiple tasks can be ran at the same time. For example, a single-core CPU can run many tasks concurrently by using a time quantum and running tasks for a certain amount of time. This means this OS is concurrent without being parallel.

4.17 The output at LINE C is CHILD: value = 5 and the output at LINE P is PARENT: value = 0. This is because the thread tid hasn't been created and executed by the time the parent reaches LINE P. However, for the child, the thread is created and ran before LINE C and because of pthread_join(), the child process must wait for thread tid to finish.

4.18

A. Because there are less kernel threads than processors, the user threads will be bottlenecked by the kernel threads not being able to leverage more processors. This means many user threads will remain idle during execution.
B. Because there are as many kernel threads as processors, the user threads will be able less constrained by the kernel threads than part A. This means few user threads will remain idle during execution.
C. Because there are more kernel threads than there are processors, some kernel threads will have to remain idle while waiting for process to become active. However, since there

are more user threads than kernel threads, user threads can be swapped with which are idle, which helps to keep user threads active.