4.7)

if there are lots of blocking calls in your program, and there are operations that can still be preformed then kernel threads will be helpful. otherwise kernel threads add a lot of overhead to your program because a new thread control block has to be made and a lot more system calls will have to be made to coordinate then, which forces more context switching. User threads make coordination between threads a lot safer, while kernel threads need to be managed a lot closer. Since its a single processor system multiple kernel threads will still have to be executed by the same processor, just like the user threads. Kernel threads are more beneficial in environments with multiple CPU's since the different kernel threads can operate on different processors while the Users threads cannot.

4.8)

Heap memory is always shared, and in most platforms global variables are shared. stack and register values are different.

4.11)

yes it is, concurrency is just when different task, have overlapping run times, where parallelism is when one task runs different sub task at the same time to more quickly finish the one main task. so you could have multiple unrelated processes running within the same time period but not actually being computed at the same time, like a context switch while one one processes is waiting for an Io operation, then a context switch back. in this example the two processes didn't run in parallel but they did run concurrently.

4.17)

could you include the figure in the pdf next time?

4.18)

(a) each thread will execute on a different core. This will have good performance since minimal context switches will be needed.

(b) threads will take turns executing on the different cores as they become available. this will have the worst performance since lots of context switches will be incurred.

(c) each kernel thread will execute on one core, and they will execute different user threads as they are available to. this will have ideal performance for a program that needs more threads than there are cores, since the context switches will be minimal and the users threads will be managed more efficiently than actual Kernel threads