Your program should accept a command line:

<program name> (-tx -ny | -p) {-f filename} i

Where x = 1 or 2. If x=1 run a threaded merge sort algorithm. If x=2 run your spinlock test, n = the number of threads, 1-4. i is the number of messages each thread will output.

If -p is given, the program will prompt the user for test 1 or 2, and the number of threads if 2 is selected.

If -f filename is given a file, named filename, is opened and the report is written to filename. If -f filename is not given the report should be written to stdout.

If anything else is input (including if a -n value is given when x = 1), output the following usage statement:

Usage: <program name> (-tx {-ny}| -p){-f filename} i

And exit the program. I would recommend creating a helper function to validate the inputs.

If the -p option is given and the user enters anything besides a 1 or a 2, output an error message and exit. If 2 is selected, prompt for the number of threads, 1-4. Again, exit if anything is entered besides 1, 2, 3 or 4.

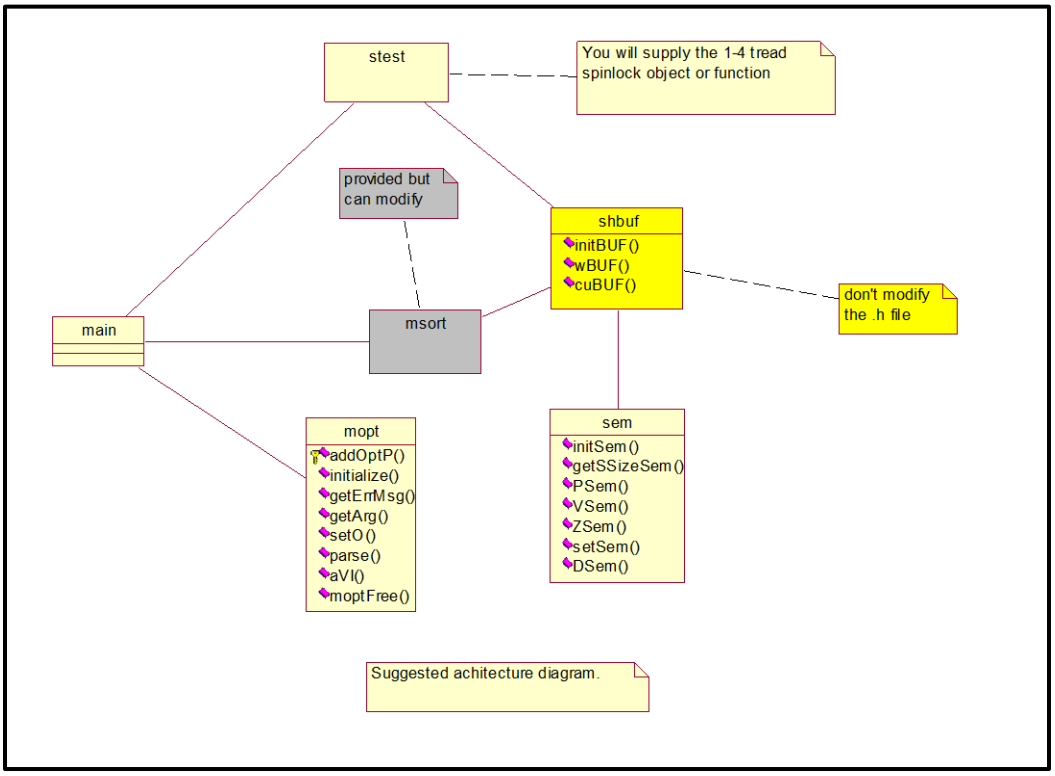
Your spinlock test will run 1-4 pthreads, which will be while(1) loops testing a value, which never changes. Periodically, each thread will send a string to the wBUF(.) function in shmem, which will time-stamp and store the messages. When cuBuf(.) is called, an informative report is output to stdout or to a file (I am intentionally leaving you a lot of leeway here). At the very least your program should output each time-stamped message in the order sent to the shbuf object with at least one summary line at the end, providing some useful information about the test run.

At this point, the program should exit.

In order to prevent thread conflict only one thread may enter the shmem object' functions at a time. You should protect access to shmem's functions with a semaphore (sem.h and sem.c supplied).

DO NOT FORGET TO REMOVE YOUR SEMAPHORE WITH THE ipcrm COMMAND.

Suggested UML-like Diagram:



You should turn in a zip file will all your .h and .c files and, optionally, a readme.txt file.