1 How To Run

To compile the program simply
run: make all
Then to run the first problem
run: problem1
Finally to run the second problem
run: list

2 Problem 1.

For the first problem we found that this was very similar to the sushi bar problem in the little book of semaphores. We followed the psudo code that they gave as a solution. We create seven threads that will run forever. In the function we have a must_wait variable that is false if less than three processes are using a resource and ture if three are. If this is set to true then then processes will be blocked in the must wait function until all 3 processes are finished. We have an if statement that checks if you are the last person eating and if the must_wait flag is set, if both are true then we unlock the mutex that is blocking the rest of the processes and allow them to go. This program will run forever but looking at the outputs will show that the program is working correctly. You will see that when there are less than three eating processes are coming and going freely. However, when there are three processes using the resources then everyone else will be waiting until the three processes have finished and then others will start to eat again.

To end this program use ctrl-c. It also may take awhile for the program to output processes coming and going freely but it shouldn't take more than two minutes.

3 Problem 2.

For this problem we first created the linked list functions that we were going to need, Insert, pop, and delete. Next we created the ligthswitch functions that the book suggested we use. These check if you are the first searcher and if you are then lock the semaphore and keep track of the number of searchers. If you are the last searcher to leave then unlock the semaphore. This is similar for the inserter as well. For our searchers function we first lock the lightswitch and then get a random value to enter. next we search through the list and if the value is found we say we found it, and if not we say we didn't find it, then we unlock the lightswitch. For inserters we first get a random number to insert and then lock our lightswitch we also block any other inserters from getting in as well. The value is then inseted into the list the size of the list is increased and we print what was inserted. The mutex blocking other inserters is unlocked and the lightswitch is is also unlocked. Finally the deleter function checks the size of the list and then from that picks a random index between 0 and size to delete. It then trys to wait on the insert and search lightswitch semaphores and can only delete if neither of them are blocking. If neither are then it waits on both causing them to block everyone else out, deletes the index, prints what was deleted, and then finally unblocks everyone. At the start of the program we create an initial list to use. we then create 10 of each type of thread and then they run. You can see for the outputs of this program that while searching is happening an insert is still able to happen but no deletes. and you will also notice that only the delete is allowed to happen and searchers and inserters aren't allowed to do anything.