**Version 4 Problems**

The fourth version of the alarm program consisted of a main thread which acted as a producer of alarms and an alarm thread as the consumer. The consumer thread would immediately grab the head node and point the head to its child and then sleep for the required amount of time. A problem occurs when an alarm is created that has a shorter time than the head node. The main thread will correctly put the new alarm as the head node, but the alarm thread will still be sleeping. To resolve this issue, I added a condition variable for the main thread to signal on when a new alarm is created. In the alarm thread, instead of sleeping it call the pthread timedwait function passing in the condition variable, the alarm mutex, and the sleep time. The alarm thread would check the return value of this function to see if it had been signaled or if the wait time had passed. If the function timed out, then the earliest alarm had finished correctly, and so the thread displayed the alarm time and looped back to the beginning. However, if the condition was signaled the alarm thread would create a new alarm with the same values as the previous head node (the node that the alarm thread 'pops' off at the beginning of the loop) and inserts it just as the main thread would. It then 'pops' off the new head node and loops back to call the timedwait function with its new sleep time. This allows for multiple earlier alarms to be inserted.

**Test Cases**

To test the problems existing in the fourth version, I entered "20 sec", "10 sec", and "5 sec" in that order. I observed that after 20 seconds, the program sounded for all three alarms at once. Instead, the program should sound them in reverse order (assuming the commands are entered quick enough). I used the same test case after updating the program. The result was as expected. I also ran other test cases where only one alarm needed to be inserted at the beginning to test that the order was kept correctly (i.e. "5 sec", "10 sec", "2 sec") and the results were correct.