**EECS3221 Assignment 2 Report**

Implementation of assignment requirements:

This portion of the report will explain in detail how each of the requirements in the assignment document were properly addressed and implemented in our program.

A3.1: In addition to the alarm thread created in the default program provided, we also created the two additional threads d1\_thread and d2\_thread which are of the same type as the alarm thread (pthread\_t). These two threads are initialized exactly the same as the alarm thread in our main function, and our status variable is checked after each initialization in order to ensure that the threads were created without any errors.

A3.2: On line 240 of our program we first check that the input is correct else we send an error message to stderr. If the command entered is valid, we enter the code block starting on line 244, where the command is processed and the required message is printed on line 250.

A3.3: At this point in execution our alarm thread is running and checking if there is an alarm to be processed. If there is no such alarm the thread’s while loop sleeps for 1 second. Otherwise, the alarm is received and we check to see if the alarm expiry time is (a) closer to an odd integer number (on line 72) or (b) closer to an even integer number (on line 85).

A3.3 (c): In both cases, the set of instructions are practically the same, but we pass the alarm to display thread 1 in the case of (a) and display thread 2 in the case of (b), where we first display the required message for the respective display thread.

A3.3 (d): Because of the while loop in the alarm thread, the alarm thread is still free to process additional incoming alarms that are sent to it.

A3.4: Similar to the alarm thread, both display thread functions loop continuously to ensure that any alarm requests sent from the alarm thread are processed properly, and on line 132 of display thread 1 and line 179 of display thread 2 we call pthread\_cond\_wait which waits for an alarm request from the alarm thread.

A3.4 (a): Both display threads perform essentially the exact same instructions. The proper receive message is displayed upon the display thread receiving the alarm request.

A3.4 (b): The display threads repeatedly display the status of the thread and sleep(2) is used to ensure that the message is only displayed every 2 seconds. The condition of the while loop checks that the alarm time has not expired.

A3.4 (c): Immediately after the while loop terminates meaning the alarm request has finished processing in the display thread, the appropriate display thread expiry message is displayed indicating that the display thread is now free to process more alarm requests. The alarm\_mutex is unlocked at this point using pthread\_mutex\_unlock.