**Part 1Q1**

The sequence of steps is as follows:

1. Process 0 sets flag[0] = true; and then process 1 runs, and sets flag[1] = true.
2. Process 0 goes to the while loop and is looping inside the while loop
3. Process 1 goes to the while loop and is looping inside the while loop

A deadlock happens here because neither process gets past the while loop and they both keep looping forever in the respective while loops.

**Part 1Q2**

This is strict alternation, and can cause starvation if one process is slower than the other.

1. Let turn = 0 at the beginning
2. Now, let P1 be far away from it’s critical section, and P0 close to critical section
3. Since turn == 0, P0 executes critical section and sets turn to 1
4. P1 hangs outside it’s critical section and never gets to the critical section
5. Turn is never set to 0, and P0 never gets to execute it’s critical section again.

Alternatively,

1. Turn = 0 to start
2. Now, P0 is far away from critical section, and P1 is near critical section
3. P0 hangs in non critical section before the while(turn!=0); line
4. Turn is never set to 1
5. P1 never gets to critical section

This violates one of the rules of mutual exclusion (an element outside the critical section blocks an element trying to execute it’s critical section) , and does not fulfil mutual exclusion.

**Part 2**

The solution to this problem is implemented as the Third Reader Writer Problem, which maintains an order of the requests in order to make sure neither the reader nor the writer is starved (in the solution given in the textbook, there is the possibility of writer starvation)

In this implementation, there are three mutexes, the order (to maintain the queue) the “mutex” (to maintain the reader count) and the mutex to control access to the database.

The reader count is in shared memory (as an integer pointer) and all the processes update it.

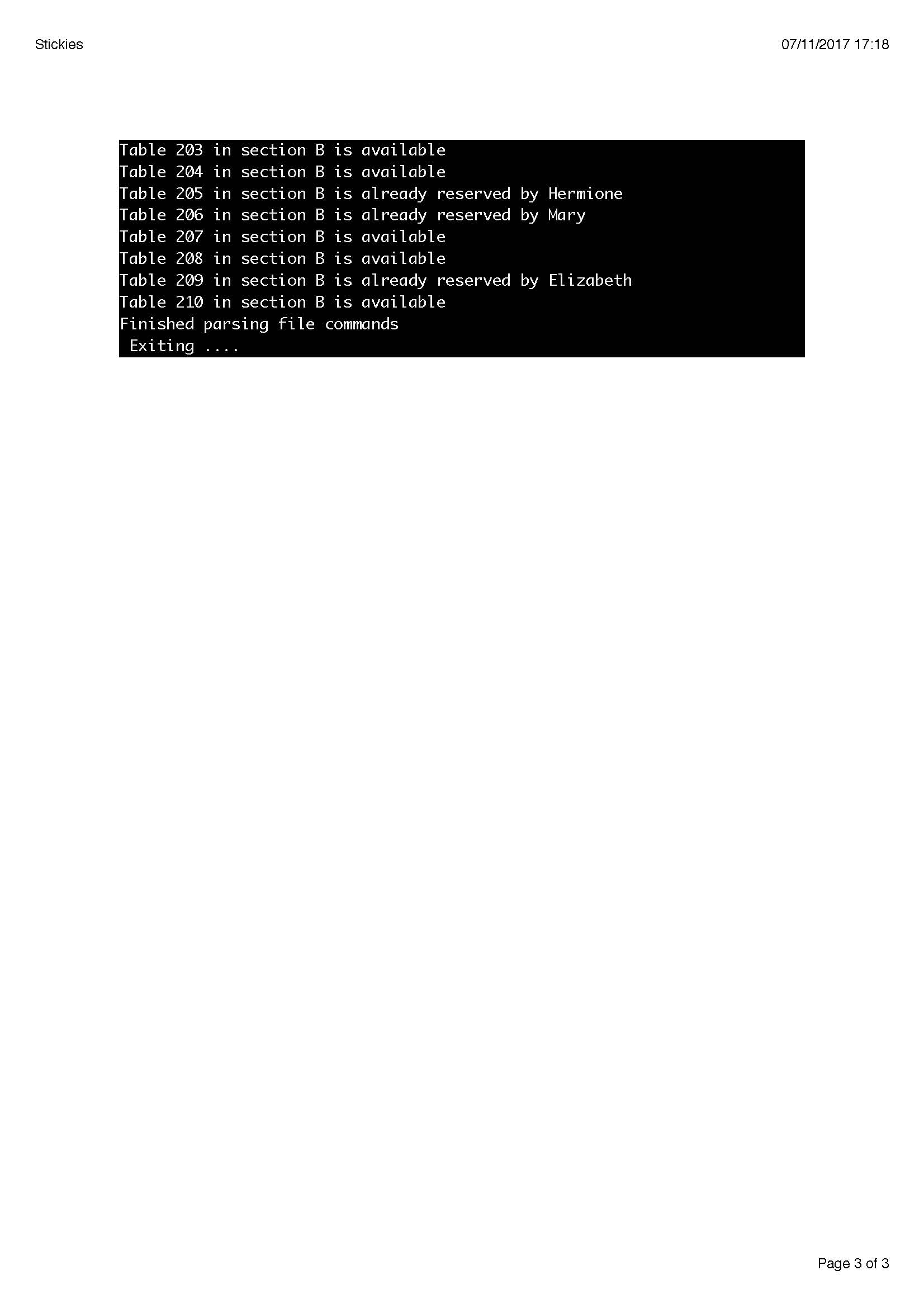
The database is implemented as a struct containing arrays of the names, tables and availability.

The trace from the output is given below, when the processes are run concurrently on two shells with two files (file2.txt and file1.txt respectively). Here the times are printed out in order to see that the two processes are accessing the database concurrently (these print statements are now commented out in the code, to make sure no extra statements are printed). These print statements are lines and , and can be uncommented to get access times and verify the output shown below:

The first process:







The second process:







The two text files are file1.txt and file2.txt and another set of outputs is attached in file1output.txt and file2output.txt