Konrad Schultz V00761540 CSC360 2.1

1. There will be one thread for each train loading, and a main controller thread that keeps track of timings and prints to stdout.

2. There is an overall controller thread that keeps a queue of train departure order. This follows the manager-worker threaded program design pattern.

3. There will be one mutex to lock the main track (current_train) and one mutex to lock the train list. (may split train list into multiple train lists which each require a mutex to have less threads accessing the same piece of data).

4. No the main thread will not be idle. It will be responsible for queueing the trains and sending them on their way. It will also spawn a thread for each train and read the input file.

5. Stations will be represented by a linked list. The main thread will be responsible for inserting based on priority order (o(n)) and sending the next available train on the track (o(1)).

6. I can think of going about this two ways, either each of the threads will have access to the list and it will be guarded by a mutex/convar or the thread will return some sort of ready signal to the main controller thread that will add it to the list of trains (I'll probably use the former).

7. From my understanding of conditional variables I don't think I will require any. Thread locking / mutual exclusion should be all the logic I require. The only conditions will be MAIN_TRACK_STATUS and TRAIN_LIST_STATUS which will be either locked or unlocked, and a thread should wait until it's unlocked and then perform its action.

8. In 15 lines or less briefly sketch the overall algorithm you will use.

```
Create all relevant mutexes and data structures
Read input file and spawn a thread for each train.
Sleep for however long the train needs to load.
Once a train is loaded display output to screen and acquire list mutex
Add train to the proper place in the list based on its priority and other factors
Unlock the list mutex
Whenever there are trains in the list and the track is empty lock the track mutex and the list mutex
Remove the train from the list and unlock the list mutex
Put it on the track for its crossing time
Unlock the track mutex
Exit train thread
At the end of main wait for all threads to exit, this ensures all trains have crossed
```