Konrad Schultz V00761540 CSC360 2.2
Changes in red

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

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 (main_track) and one mutex to lock each 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). (eastbound_lock) (westbound_lock)

4. No the main thread will not be idle. It will be responsible for dispatching 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 each train thread will be responsible for inserting itself based on priority order (o(n)) and the main thread will be responsible for 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). 3 mutexes, 1 condition variable. The dispatcher waits for something to be inserted when there is nothing in either list (condition variable) and the eastbound and westbound lists have a mutex. added_cv is associated with main_track_lock.

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. I used 1 condition variable (added_cv) to tell the dispatcher to wait while there is nothing in either list. The train threads signal this condition variable whenever something is added.

8. In 15 lines or less briefly sketch the overall algorithm you will used.
   Create all relevant mutexes and data structures
   Read input file and spawn a thread for each train.
   Sleep for train's loading time
   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
   Signal dispatcher
   •   Lock the track mutex
   •   If nothing in either list wait to be signalled
   •   Lock both list mutexes
   •   Pop the next train to go off its respective list
   •   Unlock list mutexes
   •   Sleep for train's crossing time
   •   Unlock the track mutex
   •   REPEAT UNTIL ALL TRAINS CROSSED