Assignment 3

1. Your program's parallelisation strategy and design

The main parallelization strategy is to distribute the number of train stations among the processes. Process 0, the root would always carry the start and end stations of every line in order to make spawning trains easier while the other train stations were distributed among the remaining processes.

When a process has a train that is departing from its station to another station managed by another process, it will then send a message over to that process. In order to speed things up, we use MPI_Iprobe() while receiving the message, so that the process is not struck always waiting to receive messages.

2. how deadlocks/race conditions are resolved in your implementation

MPI_Barrier() is the main way deadlocks and race conditions are resolved.

The program has 4 main steps

a. spawn trains
b. process local updates for train stations
c. send trains to other process
d. receive trains from other processes

MPI_Barrier is used after step b and c to allow all processes to process and send the trains that needs to be sent to other processes, before the processes start accepting incoming trains.

3. the key MPI constructs used to implement this strategy and why they are used

For distribution of information among the processes, like distributing an hashmap mapping the station index to mpi rank, I used MPI_Bcast to do this for every process to broadcast its own local mapping to all other processes.

I mainly used MPI_Send and MPI_Recv in my code to send and receive information, alongside MPI_Iprobe() to try to speed things up.

As there were only 3 train lines, I opted to just use MPI_world_comm for communication groups, instead of trying to create and manage 3 different MPI Communication groups for each of the train lines.