Name:

Stage	1	2	3	4	Sum
Points	5	4	8	4	21
Result					

L3: Trains

On this laboratory your have to implement train simulator program. Every train can use railroad network kept in shared memory. Trains can run independetly from each other and there can be only one train at a single track (in single direction) There are two programs in reposiotry: `sop-railway` and `sop-train`.

`sop-railway` is responsible for creation and managment of railway network. It can read a file with network description and save topology to shared memory. Additionally it has option to print current state of all connections.

`sop-train` is responsible for train simulation. It assumes that shared memory object exists, otherwise it should exit with error. A train has some speed and name. Before departure it has to find a path between two stations in network.

To help you to implement there is example railway topology in `exmaple.rail` file visible on figure 1.

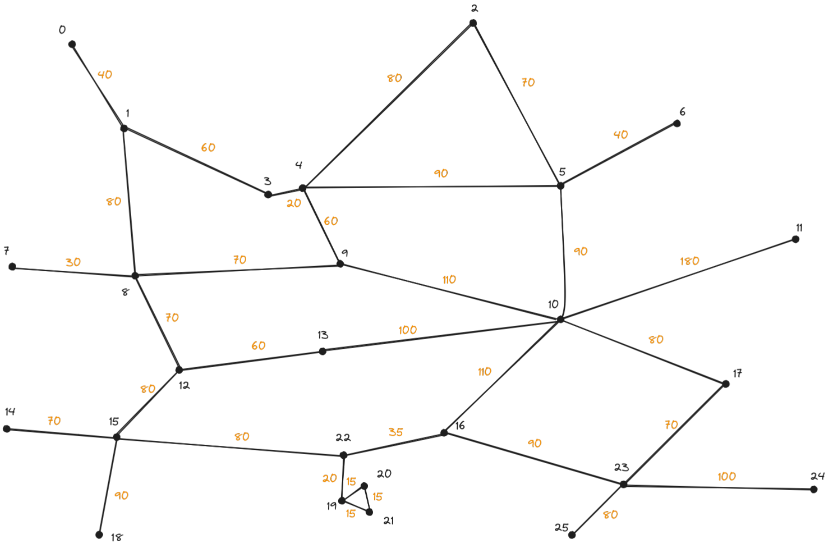

Figure 1: Topology of railway network from `example.rail`

Stages:

1. 5 p. Implement `railway_network_init` and `railway_network_destroy` functions. Use them in main function of `sop-railway`. Make sure there is no partial initialization of shared memory.

2. 4 p. Implement all options of `sop-railway` desribed in usage function. Use existing `create_railway` function to read railway data from file. Implement `railway_network_open` and `railway_network_close` and use it in program. **Hint:** use `getopt()`.

3. 8 p. Implement logic of train in `sop-train`. During the cruise train should:

    1. Lock the rail where it tries to enter (`railway_lock_connection_mutex`).
    2. Leave its name in occupation array.
    3. Sleep for x/v seconds, where x - length of rail and v - velocity of train.
    4. Remove its name from occupancy array and release lock.

    To receive path between stations use `railway_network_find_shortest_way` function. **Hint:** for calculating indices of connections and stations from each other take a look at conversion functions.

4. 4 p. Sometimes train can derail after aquiring a connection lock (it should be implemented with `exit` function). Implement derailing logic and react for arriving at destroyed connection. If train want to enetr destroyed railway it has to find new path to traget. Update `railway_lock_connection_mutex` for handling orphaned **robust mutex**. **Hint:** `railway_network_find_shortest_way` function can avoid destroyed connections after proper implementation of `railway_lock_connection_mutex` function.