Lab Exercise 7
Freight Cars

Suppose a private railway company, called PRC, wants to offer logistic services. The company has a large fleet of self-powered freight cars (goods wagons/carriages) and its own railway lines. The company is going to serve several cities in India, with a motto of fast delivery of parcels of a particular (fixed) dimensions. Customers can book parcels from any one city (origin) to a destination city. At each city, the parcels are dumped systematically in a warehouse so as to have a separate pile of parcels for each destination city, ordered based on the arrival time of the parcel. The maximum loading capacity of a freight car is 5 parcels. As soon as 5 parcels to a destination city are received, these 5 parcels are loaded on a freight car and the freight car is labelled with the destination city. An onboard computer on the freight car computes the shortest path (using Breadth First Search) to the destination city and figures out the first link (graph edge/ railway line) to proceed on. However, a freight car has a high breakdown probability, so, the freight car is not allowed to proceed on the identified link (say e) unless there are at least 5 freight cars which all need to proceed on the same link e. Moving at least 5 freight cars in the form of a train gives some fault tolerance, since assuming that if some of the cars in a train break down, there will be other cars which provide the pull/push to move along the track. Therefore, at least 5 cars are required to group together to make a train that can move along a link.

In case a car is ready to move along a link, but there are not enough cars (minimum 5) that are ready to move at that time tick t along the same link, then the onboard computer on the car performs a depth first search and works out a path to the destination city and figures out the first link to proceed along the DFS path. If there are still not enough freight cars to move to the first link on the DFS path, then the car waits at the origin city for the next time tick and tries again to form a group.

When a train of minimum 5 freight cars traverses a link (it takes 1 time step to traverse a link) and reaches the opposite station (vertex), the onboard computer again works out the BFS and DFS path to the destination. If minimum 5 freight cars are available to move along an identified link (which is the first link of their BFS or DFS path), then they will form a train and proceed along the identified track. Note that a train can have more than 5 freight cars also. We only need minimum 5 cars to make them move on a link.

At a given station (vertex) there can be several freight cars arriving from the incident edges (links/ tracks). Also, there will be freight cars that have been loaded with parcels at that station itself. A proper reorganisation of the freight cars will take place at each time tick at the current station so that the movement along the subsequent link is planned.

Assumptions:

1. There are enough freight cars at each station so that you can load them with parcels. However, a car will move only if it has exactly 5 parcels and there are at least 5 other cars to share the same link on which this car needs to move.
2. Each edge can be assumed to be a double track, so parallel movement can happen for several trains along to and fro directions.

Task 1: Reading Input to the program

The program reads two input files **graph.txt** and **bookings.txt**

The graph.txt file contains the list of links between station codes. Each graph edge is formed by a pair of stations/vertices. For example

JODHPUR JAIPUR
JODHPUR DELHI

```
JODHPUR    AHMEDABAD
JODHPUR    JAISALMER
JAIPUR     UDAIPUR
DELHI      KOLKATA
DELHI      CHENNAI
DELHI      MUMBAI
JAIPUR     JABALPUR
JODHPUR    INDORE
```

The booking.txt file lists the bookings of parcels received at various cities. A city code is made of capital letters and can be up to 20 characters long, without any blank space or special characters. [Time tick] and [Priority] are integers. A higher integer used for [Priority] indicates a higher priority. [Parcel ID] is an alphanumeric string used to uniquely identify a parcel.

[Time tick], [Parcel ID], [Source city code], [Destination city code], [Priority]

```
1   P1  JODHPUR   MUMBAI   10
1   P2  JODHPUR    AHMEDABAD    9
2   P3  JODHPUR    JAIPUR    1
2   P4  JODHPUR    JAIPUR    2
2   P5  JAIPUR     MUMBAI    2
3   P6  JAIPUR     MUMBAI    10
```

Task 2: Forming the undirected graph

The graph.txt file will be read to form a graph G with the cities as vertices and edges linking the cities with a direct rail line connection. The graph can be represented in the form of an adjacency list. The neighbours of a given vertex as mentioned in the adjacency list are to be ordered following the same order in which they are encountered in the file graph.txt. The graph G will be used for performing breadth first traversal and depth first traversal starting from any given vertex and work out a path to a given destination city/vertex. At each vertex, the DFS and BFS paths to every possible destination vertices can be computed once and stored. A freight car at a given vertex can refer to the computed paths to find the next link to proceed onto in order to reach a given destination. However, a freight car is allowed to traverse a unexplored edge every time. That means, It should not retrace back on an edge it has already traversed. To avoid retracing back on an edge that was already visited, the onboard computer can compute an alternative DFS path for the freight car that can avoid the set of links already traversed. In case all the DFS paths have some edges that were already visited by the freight car, then it (freight car) will remain stranded at that city.

Task 3: Processing the parcel bookings

The parcel bookings come at a time tick. All the processing related to a parcel, such as, receiving the booking, piling it up, putting it into a freight car, computing the link on which the freight car can proceed, and connecting together multiple freight cars that plan to proceed on the same link, can be done in a single time tick. Movement of a freight car from one vertex to another vertex of the graph takes 1 time tick. So, if 5 parcels are received at time tick t at a given origin city to a given destination, then all the 5 parcels can be put into a freight car at time tick t. If 5 freight cars are identified which are all going to use the same link, they can proceed to the next city (vertex) at time tick t and reach there at time tick $t + 1$.

Your program should have an outer while loop which will increment the time ticks. In each time tick, states will be updated for
- all the parcels that have been received till that time tick
- the freight cars that are in use
- all the stations (vertices). A vertex will keep information about what all freight cars have been loaded at that station or arrived at that station from other stations, or departed from that station.

At any time tick t, if a high priority parcel comes in, then it will be shipped out in the first bunch of 5 freight cars that will leave on or after time tick t. (Use a max heap to order the parcels at a given station).

However, once a freight car leaves its origin station, the priorities of the parcels in it will not be used subsequently and will have no significance.

Task 4: Reporting the simulation result

After processing all the parcels mentioned in the file booking.txt, the simulation will continue for some more time ticks till it converges. At convergence, none of the freight cars, parcels or vertices are going to update their states with passage of time.

Your software should report the following:
1. The path taken by each parcel till convergence. The path should be the sequence of stations and the time tick at which a station on the path is reached.
2. The list of stranded parcels, which have not been able to reach the destination. The stranded parcels may be at the origin vertex, or any non-destination vertex (if the freight car containing them is not able to move out).

Class templates for the following classes will be provided.

PRC

All the necessary information that supports the operations of the Private Railway Company will be there in this class. For example:
- Graph representing the rail network.
- List of pointers to all freight car objects.
- List of pointers to all parcels that have been booked.
- Minimum number of freight cars that are allowed to move out of a station in the form of a train. In our writeup we have been indicating this number as 5. However, this is a data field of the class PRC. You can experiment by changing this value.
- The maximum number of parcels which can be accommodated inside a freight car. In our writeup we have been indicating this number as 5. However, this is a data field of the class PRC. You can experiment by changing this value.
- Time tick t

Graph

This class supports methods for the Graph. The graph is represented as an adjacency list of all the stations (vertices). An edge is represented as an unordered pair of vertices.
Methods supported by the graph:
- Find BFS path from a given source to a destination vertex
- Find DFS path from a given source to a destination vertex
- Find DFS path from a given source to a destination vertex, while avoiding certain edges (links).

Vertex

This class supports all the operations that need to be carried out at a station (vertex of the rail network). Example methods:

- *groupFreightCars():* Group the freight cars that share the same outbound track. If adequate number of freight cars are not available for movement, it will attempt to group based on the alternative option of the outbound link (as proposed by DFS path search).
- *moveTrainToLink ():* moves a train of 5 or more freight cars on an outbound link. The effect of this procedure is that the cars will be at the opposite vertex on the next time tick. The cars will no longer be available at this vertex.
- *loadParcel (parcel object):* If the existing freight cars are full, this method will instantiate a new freight car and load the parcel into it. This method also needs to take care of the priority of the parcel while loading it. If a freight car is not having 5 parcels, it cannot move out. A high priority parcel needs to be appropriately loaded so that it ships before the low priority parcels. All parcels with the same priority level will be shipped out in First booked first out order. [Think !]

FreightCar

This class supports methods for inserting the parcels into the freight car and labelling the freight car with the destination city of all the parcels inside it. It keeps track of the next link on which it needs to be moved.

Parcel

This class is used to instantiate parcel objects with data fields such as parcel ID, origin, destination and priority.

Assumptions:

1. PRC does not consider the actual distance between one vertex to another while computing the shortest path. The shortest path is the one which has the minimum number of links in it.

2. PRC is going to use a third party software to plan the appropriate way to shunt (reorganize) the freight cars so that those going to a particular outbound link can be grouped together. So your code need not address this aspect. You can conveniently group together the cars that need to proceed on a given outbound link.