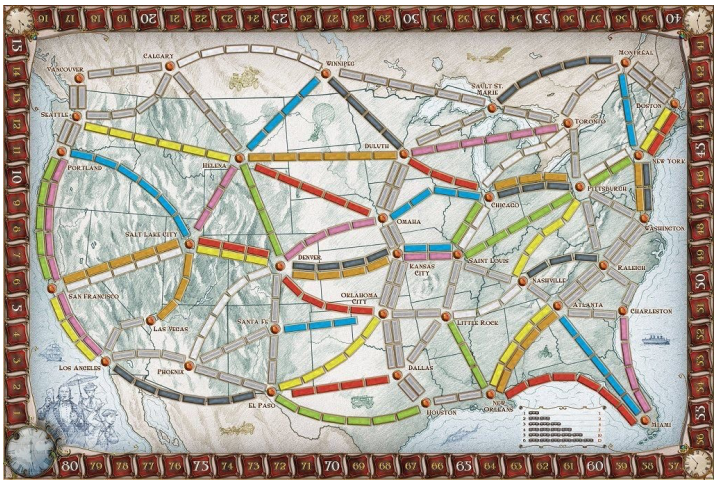

Figure 1: https://meepletown.com/wp-content/uploads/2011/03/TTR.jpg (Retrieved on February 12, 2019)

Note: This specification is provided as part of Assignment 1 of CSCI5308 in Summer 2021 and explains what the whole code is supposed to do. However you are not going to work with the work code, nor build the whole solution. Please see the assignment instructions for your task, e.g. writing unit tests for the code. Do not spend time writing code for the ticket to ride problem.

Background: Ticket to Ride, the Board Game

In a popular board game, called "Ticket to Ride"[1], the goal of the game is to build a rail network that covers the routes each player is given. A route is specified by the end-point cities, and is constructed by building segments illustrated on the game board above. For example, to build a route between Boston and Winnipeg, a player may choose to build the segments: Boston to Montreal, Montreal to Toronto, Toronto to Duluth, and Duluth to Winnipeg. The longer the segments the more expensive they are to build, and routes with more segments take longer to build. Thus, it is in the player's interest to build the routes she is allocated in the most efficient way possible.

Problem: Minimize Network Length

Given a game board of rail segments and a list of routes, your task is to compute the total cost of building a network of prescribed routes, assuming that the shortest distance for each route is chosen.

[1] Published by Days of Wonder

For example, given the following game board and routes: The resulting cost computation would be:

```
Boston 2 Montreal
Boston 2 New_York
Chicago 4 Toronto
Chicago 3 Pittsburgh
Montreal 3 New_York
Montreal 3 Toronto
New_York 2 Washington
New_York 2 Pittsburgh
Pittsburgh 2 Toronto
Pittsburgh 2 Washington
done
Washington Montreal
Chicago New_York
done
```

Figure 2: Sample of possible rail segments and two routes.

```
The rail network consists of:
  Chicago 3 Pittsburgh
  Montreal 3 New_York
  New_York 2 Pittsburgh
  New_York 2 Washington
The total cost is: 10
```

Figure 3: Segments and total cost of building a rail network for the specified routes and game board in Figure 2

The idea is to have a program that reads a game board and routes, and computes which segments should be constructed and the total cost. Write a program called RouteCost.java that reads in a game board and routes from the console (*System.in*) and outputs the set of segments to be built and the total cost.

Input

Your program should read in the input using a *Scanner* object, which is instantiated with *System.in*. The input will comprise two sections with one or more lines in each section. The first section contains the game board and comprises zero or more lines of the form

C_1 *L* C_2

where C_1 and C_2 are the end-points of a segment on the game board and *L* is the length of the segment. E.g., "Toronto 3 Montreal". The section is terminated by a single word "done".
The second section contains the routes and comprises zero or more lines of the form

C_1 C_1

where C_1 and C_2 are the end-points of a route, comprising one or more segments. E.g., "Montreal Washington". The section is terminated by a single word "done". Hint: All you need to use are the next() and nextInt() methods of the *Scanner* object.

Semantics

The game board is connected and all the city names are single words, e.g., "Las_Vegas". You may assume that all game boards and all routes will be valid. All routes will have distinct end-points (no cycles or 0-length routes). The segments are bidirectional, i.e., can be used in either direction, and the game board represents a weighted undirected graph. Routes may intersect and may share segments. Segments need only be counted once though. The cost of a route is the sum of the lengths of the segments in the route. A rail network is considered minimal if each route has minimum cost.

Output

Your program should output to *System.out*. Each line should be terminated by a new line character. The output should begin with the line:

The rail network consists of:

followed by the list of segments used in the rail network. Each segment should be indented two (2) spaces, and the segments should be in sorted order, where the (C_1,*L*,C_2) precedes (C_1', L', C_2') if C_1 lexically precedes C_1', or if C_1 equals C_1', then C_2 must lexically precede C_2'.
The format of the segments is the same format as the input.

The list of segments should be followed by the line

The total cost is: T

where T is the sum of lengths of the segments. See Figure 3 for an example.

Example

Sample Input	Sample Output
Charleston 2 Raleigh Chicago 3 Pittsburgh Chicago 2 Saint_Louis Chicago 4 Omaha Chicago 3 Duluth Dallas 2 Little_Rock Dallas 2 Oklahoma_City Denver 4 Omaha Denver 4 Kansas_City Denver 4 Oklahoma_City Denver 2 Santa_Fe Denver 3 Salt_Lake_City Duluth 2 Omaha Duluth 6 Helena Helena 4 Winnipeg Helena 5 Omaha Helena 4 Denver Helena 3 Salt_Lake_City Kansas_City 2 Saint_Louis Kansas_City 2 Oklahoma_City Kansas_City 1 Omaha Las_Vegas 3 Salt_Lake_City Little_Rock 3 Nashville Little_Rock 2 Oklahoma_City Little_Rock 2 Saint_Louis Nashville 3 Raleigh Nashville 2 Saint_Louis Nashville 4 Pittsburgh New_York 2 Washington New_York 2 Pittsburgh Oklahoma_City 3 Santa_Fe Pittsburgh 2 Washington Pittsburgh 2 Raleigh Pittsburgh 5 Saint_Louis Raleigh 2 Washington Saint_Louis 2 Chicago done Denver Washington Chicago Oklahoma_City done	The rail network consists of: Chicago 2 Saint_Louis Denver 4 Kansas_City Kansas_City 2 Saint_Louis Little_Rock 2 Oklahoma_City Little_Rock 2 Saint_Louis Nashville 3 Raleigh Nashville 2 Saint_Louis Raleigh 2 Washington The total cost is: 19

Hints and Suggestions

- Use a 2-phase algorithm: Create a weighted graph representing the game board. Then, use Dijkstra's shortest weighted path algorithm to find the shortest routes.
- The sample solution is under 200 lines of code.
- Your code must be well commented and indented. Please see the Assignments section for this course on Brightspace for Code Style Guidelines.
- You may assume that all input will be correct.
- Be sure to test your programs using the provided tests or *Mimir*.