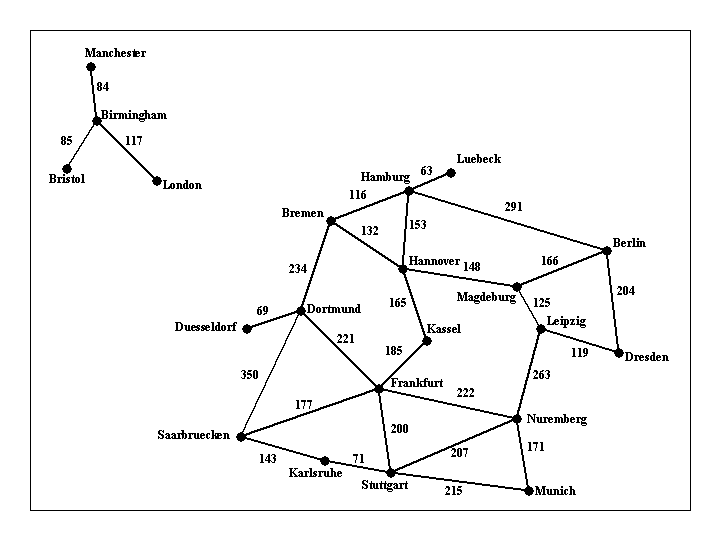
**Uninformed Search & Informed Search**

**Task 1**

***Note: This is a ABET Assesment Task***

  
Figure 1: Visual representation of [input1.txt](http://crystal.uta.edu/~gopikrishnav/classes/2021/spring/4308_5360/assmts/assmt1_files/input1.txt)

Implement a search algorithm that can find a route between any two cities. Your program will be called find\_route, and will take exactly commandline arguments as follows:  
  
***find\_route input\_filename origin\_city destination\_city*** ***heuristic\_filename***  
  
An example command line is:  
  
find\_route input1.txt Bremen Kassel (For doing Uninformed search)  
or  
find\_route input1.txt Bremen Kassel h\_kassel.txt (For doing Informed search)  
  
If heuristic is not provided then program must do uninformed search. Argument input\_filename is the name of a text file such as [input1.txt](http://crystal.uta.edu/~gopikrishnav/classes/2021/spring/4308_5360/assmts/assmt1_files/input1.txt), that describes road connections between cities in some part of the world. For example, the road system described by file input1.txt can be visualized in Figure 1 shown above. You can assume that the input file is formatted in the same way as [input1.txt](http://crystal.uta.edu/~gopikrishnav/classes/2021/spring/4308_5360/assmts/assmt1_files/input1.txt): each line contains three items. The last line contains the items "END OF INPUT", and that is how the program can detect that it has reached the end of the file. The other lines of the file contain, in this order, a source city, a destination city, and the length in kilometers of the road connecting directly those two cities. Each city name will be a single word (for example, we will use New\_York instead of New York), consisting of upper and lowercase letters and possibly underscores.  
  
**IMPORTANT NOTE**: MULTIPLE INPUT FILES WILL BE USED TO GRADE THE ASSIGNMENT, FILE [input1.txt](http://crystal.uta.edu/~gopikrishnav/classes/2021/spring/4308_5360/assmts/assmt1_files/input1.txt) IS JUST AN EXAMPLE. YOUR CODE SHOULD WORK WITH ANY INPUT FILE FORMATTED AS SPECIFIED ABOVE.  
  
The program will compute a route between the origin city and the destination city, and will print out both the length of the route and the list of all cities that lie on that route. It should also display the number of nodes expanded and nodes generated. For example,  
  
find\_route input1.txt Bremen Kassel  
  
should have the following output:  
  
nodes expanded: 12  
nodes generated: 20  
distance: 297.0 km  
route:  
Bremen to Hannover, 132.0 km  
Hannover to Kassel, 165.0 km  
  
and  
  
find\_route input1.txt London Kassel  
  
should have the following output:  
  
nodes expanded: 7  
nodes generated: 7  
distance: infinity  
route:  
none  
  
For full credit, you should produce outputs identical in format to the above two examples.  
  
If a heuristic file is provided then program must perform Informed search. The heuristic file gives the estimate of what the cost could be to get to the given destination from any start state (note this is just an estimate). In this case the command line would look like  
  
find\_route input1.txt Bremen Kassel h\_kassel.txt  
  
Here the last argument contains a text file what has the heuristic values for every state wrt the given destination city (note different destinations will need different heuristic values). For example, you have been provided a sample file [h\_kassel.txt](http://crystal.uta.edu/~gopikrishnav/classes/2021/spring/4308_5360/assmts/assmt1_files/h_kassel.txt) which gives the heuristic value for every state (assuming kassel is the goal). Your program should use this information to reduce the number of nodes it ends up expanding. Other than that, the solution returned by the program should be the same as the uninformed version. For example,  
  
find\_route input1.txt Bremen Kassel h\_kassel.txt  
  
should have the following output:  
  
nodes expanded: 3  
nodes generated: 8  
distance: 297.0 km  
route:  
Bremen to Hannover, 132.0 km  
Hannover to Kassel, 165.0 km 

**Suggestions**

Pay close attention to all specifications on this page, including specifications about output format, submission format. Even in cases where the program works correctly, points will be taken off for non-compliance with the instructions given on this page (such as a different format for the program output, wrong compression format for the submitted code, and so on). The reason is that non-compliance with the instructions makes the grading process significantly (and unnecessarily) more time consuming.