Building A Graph, Reading

The most versatile way to build a database of nodes with their connections is to read them from a file. A parsing file is included in this reading for you to copy, paste, and mark up for your use.

The parsing is for an external database in the form of a "flat file". The file has three lines for each connection -- two with city names and one blank line (so that the text editing of the file will be easier). To end the input, there's just one line with EOF on it where the first city name would have been.

There is no separate list of cities -- the list is build as the names of cities are discovered as the connections are processed. For example, if the first connecting cities are San Francisco and Sacramento, the parsing program creates nodes for both of those cities, and lists the other as a neighbor in both.

It's important that the names of the cities be spelled and cased the same way when they appear in multiple places in the file, or else they will be seen as different cities.

Here is a complete working program:

```
#include <fstream>
#include <iostream>
#include <list>
#include <string>
#include <vector>
using namespace std;

struct Node
{
  string name;
  list<int> neighbors;
};

int main()
{
  ifstream fin;
  fin.open("cities.txt");
  if (!fin.good()) throw "I/O error";

  // process the input file
  vector<Node> database;
  while (fin.good()) // EOF loop
  {
    string fromCity, toCity;

    // read one edge
    getline(fin, fromCity);
    if (fromCity == "EOF") break;
    getline(fin, toCity);
    fin.ignore(1000, 10); // skip the separator

    // add nodes for new cities included in the edge
    int iToNode = -1, iFromNode = -1, i;
    for (i = 0; i < database.size(); i++) // seek "to" city
      if (database[i].name == fromCity)
```

```
      if (database[i].name == fromCity)
        break;
    if (i == database.size()) // not in database yet
    {
      // store the node if it is new
      Node fromNode = {fromCity};
      database.push_back(fromNode);
    }
    iFromNode = i;

    for (i = 0; i < database.size(); i++) // seek "from" city
      if (database[i].name == toCity)
        break;
    if (i == database.size()) // not in vector yet
    {
      // store the node if it is new
      Node toNode = {toCity};
      database.push_back(toNode);
    }
    iToNode = i;

    // store bi-directional edges
    database[iFromNode].neighbors.push_back(iToNode);
    database[iToNode].neighbors.push_back(iFromNode);
  }
  fin.close();
  cout << "Input file processed\n";
}
```

Note that this code assumes that all connections are *bi-directional*. For graphs with one-way connections, that code block can be modified.