(6) [15 points]

You will be given a directed graph in which nodes are described by their names (strings), and you are supposed to read in the graph structure fast. More specifically, Ms. Trojan already implemented for you the following class:

```
class GraphNode {
   public:
      GraphNode (const string & name);
             // generates a new node without edges, with the given name
             // runs in time O(1)
      void addEdgeTo (GraphNode *otherEndpoint);
             // adds an outgoing edge from this node to otherEndpoint
             // runs in time O(1)
      void addEdgeFrom (GraphNode *otherEndpoint);
             // adds an incoming edge to this node from otherEndpoint
             // runs in time O(1)
      const string & getName () const;
             // returns the name of the node in time O(1)
      // other private and public parts won't be relevant here
};
```

You will be asked to read a directed graph in the following format: the first line contains n and m, the number of nodes and the number of edges. The next n lines each contain the name of one node; names will only be lowercase characters, and nothing else (in particular, no spaces). The next m pairs of lines each describe one edge: the first line of a pair will be the name of the start node of the edge, and the second line the name of the end node. Only valid node names listed above will occur.

You are to generate the graph structure, by creating all the necessary nodes, and correctly adding all the edges between them. Specifically, for each directed edge (u, v), you should add v among the outgoing edges of u, and u among the incoming edges of v.

For full credit, your solution should run in time $O((m + n) \log(m + n))$. If your solution is slower than that, you will get almost no credit. The code skeleton on the next page is supposed to help you. If you don't like it, feel free to cross out any of the code we gave you.

```
#include <iostream>
#include <string>









using namespace std;

int main ()
{
   int n, m;
   cin >> n >> m;










   for (int i = 0; i < n; i ++)
   {
      string nodename;
      cin >> nodename;










   }
   for (int j = 0; j < m; j ++)
   {
       string nodenameStart, nodenameEnd;
       cin >> nodenameStart;
       cin >> nodenameEnd;











   }
   // other stuff using the graph goes here - not yours to worry about
}
```