```cpp
/*
280
Graph Assignment 1
Elliot Shaw, Ben Lohman
*/

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

struct Node {
    char lable;
    vector<Node*> links;
    vector<int> weights;
};

void adjacent(const vector<Node*>& links, const char lable) {
    int i;
    for (i = 0; i < links.size() && links[i]->lable != lable; i++);
    for (int li = 0; li < links[i]->links.size(); li++)cout << links[i]->links
      [li]->lable << "";
    cout << endl;
}//adjacent

string DFSH(const Node* a, string str) {
    size_t found = str.find(a->lable);
    if (found == string::npos) {
        str += a->lable;
        for (int i = 0; i < a->links.size(); i++) {
            str = DFSH(a->links[i], str);
        }
        cout << a->lable << " ";
    }
    return str;
}//DFS helper

void DFS(const vector<Node*>& vec, const char letter) {
    int i;
    for (i = 0; vec[i]->lable != letter; i++);
    string str = "";
    str += letter;
    for (int j = 0; j < vec[i]->links.size(); j++) {
        str = DFSH(vec[i]->links[j], str);
    }

}//DFS
```

```
49  string connectedh(Node* a, string str) {
50      size_t found = str.find(a->lable);
51      if (found == string::npos) {
52          str += a->lable;
53          for (int i = 0; i < a->links.size(); i++) {
54              str = connectedh(a->links[i], str);
55          }
56      }
57      return str;
58  }//connected helper
59
60  void connected(const vector<Node*>& vect) {
61      int s = vect.size();
62      string str = "";
63      for (int i = 0; i < s; i++) {
64          str = "";
65          if (connectedh(vect[i], str).size() != s) {
66              cout << "FALSE" << endl;
67              return;
68          }
69      }
70      cout << "TRUE" << endl;
71  }
72
73  bool contains(char c, string str) {
74      for (int i = 0; i < str.size(); i++) {
75          if (str[i] == c) {
76              return true;
77          }
78      }
79      return false;
80  }//checks if char is in a string
81
82  void BFS(const vector<Node*>& vect, const char letter) {
83      int i;
84      string str = "";
85      str += letter;
86      vector<Node*> q;
87      for (i = 0; vect[i]->lable != letter; i++);
88      Node* r = vect[i];
89      for (int j = 0; j < r->links.size(); j++) {
90          q.push_back(r->links[j]);
91      }
92
93      for (int j = 0; j < q.size(); j++) {
94          cout << q[j]->lable << " ";
95          str += q[j]->lable;
96          for (int l = 0; l < q[j]->links.size(); l++) {
97              if (!contains(q[j]->links[l]->lable, str)) {
```

```
 98                 q.push_back(q[j]->links[l]);
 99             }
100         }
101     }
102 }//BFS
103
104 int main()
105 {
106     string i;
107     cout << "file name: ";
108     cin >> i;
109
110     ifstream ifile(i); //pre-req file is weighted diagraph representation
111     vector<Node*> allofem;
112     char c;
113     for (string line; getline(ifile, line); )
114     {
115         char c = line[0];
116         int ni;
117         for (ni = 0; ni < allofem.size() && allofem[ni]->lable != c; ni++);
118         if (ni == allofem.size()) {
119             Node* tmp = new Node;
120             tmp->lable = c;
121             allofem.push_back(tmp);
122         }
123         for (int i = 4; i < line.size() - 1; i += 4) {
124             int li;
125             for (li = 0; li < allofem[ni]->links.size() && allofem[ni]->links
                  [li]->lable != line[i]; li++);
126             if (li == allofem[ni]->links.size()) {
127                 char lc = line[i];
128                 int li2;
129                 for (li2 = 0; li2 < allofem.size() && allofem[li2]->lable !=
                      lc; li2++);
130                 if (li2 == allofem.size()) {
131                     Node* tmp2 = new Node;
132                     tmp2->lable = lc;
133                     allofem.push_back(tmp2);
134                 }
135                 allofem[ni]->links.push_back(allofem[li2]);
136                 allofem[ni]->weights.push_back(line[i + 2] - 48);
137             }
138             else {
139                 allofem[ni]->weights[li] = line[i + 2] - 48;
140             }
141         }
142     }
143     cout << "command file name: ";
144     cin >> i;
```

```
145
146     ifstream ifile2(i);
147     string str;
148     while (!ifile2.eof())
149     {
150         ifile2 >> str;
151         cout << endl << endl << str << " ";
152         if (str != "CONNECTED") {
153             ifile2 >> c;
154             cout << c << endl;
155             if (str == "ADJACENT") {
156                 adjacent(allofem, c);
157             }
158             if (str == "DFS") {
159                 DFS(allofem, c);
160             }
161             if (str == "BFS") {
162                 BFS(allofem, c);
163             }
164         }
165         else {
166             cout << endl;
167             connected(allofem);
168         }
169     }
170 }
```