```cpp
/*************************************************************
* PROGRAMMED BY : Daniel Olaes & Wesley Chok
* CLASS         : CS1D
* SECTION       : MW: 2:30 P.M.
* ASSIGNMENT #9 : DFS & BFS
*************************************************************

Part A - This part will demonstrate DFS starting
at Chicago (choosing the edge with the smallest
mileage) using an adjacency list.The discovery
edges and the back edges will be identified as
well.Part B - This part will demonstrate BFS
starting at Chicago (choosing the edge with the
smallest mileage) using an adjacency matrix.The
discovery edges and the cross edges will be
identified as well.*/

#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>

using namespace std;

enum cities
{
    Atlanta,
    Boston,
    Chicago,
    Dallas,
    Denver,
    Houston,
    KansasCity,
    LosAngeles,
    Miami,
    NewYork,
    SanFrancisco,
    Seattle,
    Empty
};

struct edge
{
    cities city1;
    cities city2;
    edge(cities c1, cities c2) : city1(c1), city2(c2) {};
};

struct distBtwn
{
    cities city1;
    cities city2;
    int distance;
    distBtwn(cities c1, cities c2, int dist) :
             city1(c1), city2(c2), distance(dist) {}
};

```

```cpp
const distBtwn distances[23] = {distBtwn(Seattle, SanFrancisco, 807),
                                distBtwn(Seattle, Denver, 1331),
                                distBtwn(Seattle, Chicago, 2097),
                                distBtwn(SanFrancisco, Denver, 1267),
                                distBtwn(SanFrancisco, LosAngeles, 381),
                                distBtwn(Denver, LosAngeles, 1015),
                                distBtwn(Denver, KansasCity, 599),
                                distBtwn(Denver, Chicago, 1003),
                                distBtwn(Chicago, Boston, 983),
                                distBtwn(Chicago, NewYork, 787),
                                distBtwn(Chicago, KansasCity, 533),
                                distBtwn(Boston, NewYork, 214),
                                distBtwn(KansasCity, LosAngeles, 1663),
                                distBtwn(KansasCity, NewYork, 1260),
                                distBtwn(KansasCity, Atlanta, 864),
                                distBtwn(KansasCity, Dallas, 496),
                                distBtwn(LosAngeles, Dallas, 1435),
                                distBtwn(Atlanta, NewYork, 888),
                                distBtwn(Dallas, Atlanta, 781),
                                distBtwn(Dallas, Houston, 239),
                                distBtwn(Atlanta, Houston, 810),
                                distBtwn(Atlanta, Miami, 661),
                                distBtwn(Houston, Miami, 1187)};

string enumGet(cities v);
void PrintHeader(ostream &output, string asName, int asNum, char asType);
    //Post: Outputs the class header.s
void OutputAssignmentDescription();
    //Post: Outputs the assignment description.

#include "dfs.h"
#include "bfs.h"

int main()
{
    PrintHeader(cout, "DFS & BFS", 9, 'A');
    OutputAssignmentDescription();

    cout << "\nPART A - DFS (Adjacency List):\n";
    dfsGraph d(12);
    cout << "\nDFS Traversal:\n";
    d.dfsTraversal(Chicago);
    d.printEdges();

    cout << "\n\nPART B - BFS (Adjacency Matrix):\n";
    bfsGraph b(12, 45);
    cout << "\nBFS Traversal:\n";
    b.bfsTraversal(Chicago);
    b.printEdges();
    return 0;
}

string enumGet(cities v)
{
    string str;
    switch(v)
    {
```

```
115     case Atlanta: str = "Atlanta";
116      break;
117     case Boston: str = "Boston";
118      break;
119     case Chicago: str = "Chicago";
120      break;
121     case Dallas: str = "Dallas";
122      break;
123     case Denver: str = "Denver";
124      break;
125     case Houston: str = "Houston";
126      break;
127     case KansasCity: str = "Kansas City";
128      break;
129     case LosAngeles:str = "Los Angeles";
130      break;
131     case Miami: str = "Miami";
132      break;
133     case NewYork: str = "New York";
134      break;
135     case SanFrancisco: str = "San Francisco";
136      break;
137     case Seattle: str = "Seattle";
138      break;
139     default: str = "EMPTY";
140     }
141     return str;
142 }
143
144 void PrintHeader(ostream &output, // OUT - Represents either console or out file
145                  string asName,   // OUT - The assignment name
146                  int asNum,       // OUT - The assignment number
147                  char asType)    //  OUT - The assignment type
148 {
149
150     output << left;
151     output << "*************************************************************";
152     output << "\n* PROGRAMMED BY : Daniel Olaes & Wesley Chok";
153     output << "\n* "        << setw(14) << "CLASS"   << ": CS1D";
154     output << "\n* "        << setw(14) << "SECTION" << ": MW: 2:30 P.M.";
155     output << "\n* ";
156
157     // PROCESSING/OUTPUT - If assignment type is 'L', then it will output
158     // the Lab number, otherwise, it will output the assignment number
159     if(toupper(asType) == 'L')
160     {
161         output << "LAB #" << setw(9);
162     }
163     else
164     {
165         output << "ASSIGNMENT #" << setw(2);
166     }
167
168     output << asNum << ": " << asName << endl;
169     output << "*************************************************************"
170            "\n\n";
171
```

```
172     output << right;
173 }
174
175 void OutputAssignmentDescription()
176 {
177     const int MAX_LENGTH = 50;
178     string lineStr;     // OUT  - Outputs the num of words that fit in a line
179     string outWordStr;  // CALC - Used for adding certain words to the line to
180                         //        see if they don't exceed the max chars
181     string inputStr;    // OUT  - The assignment description to be used
182     int    index;       // CALC - Used for for loop to iterate through the plot
183
184     inputStr =
185     "Part A - This part will demonstrate DFS starting at Chicago (choosing the "
186     "edge with the smallest mileage) using an adjacency list."
187     "The discovery edges and the back edges will be identified as well."
188     "Part B - This part will demonstrate BFS starting at Chicago (choosing the "
189     "edge with the smallest mileage) using an adjacency matrix."
190     "The discovery edges and the cross edges will be identified as well.";
191     lineStr    = "";
192     outWordStr = "";
193
194     int    strLength = inputStr.length();
195
196     // Word wraps the description
197     for(index = 0; index < strLength ; index++)
198     {
199         if(inputStr[index] != ' ')
200         {
201             // Add character to word
202             outWordStr = outWordStr + inputStr[index];
203         }
204         // Output line, then clear it
205         else
206         {
207             if((lineStr.length() + outWordStr.length()) > MAX_LENGTH)
208             {
209                 cout << lineStr << endl;
210                 lineStr.clear();
211             }
212             // Add to the line, clear the word
213             lineStr = lineStr + outWordStr + " ";
214             outWordStr.clear();
215         }
216     }
217
218     // PROCESSING - Since the loop ends before displaying the last word/line
219     //  we need to repeat the process and output the remaining words
220     if((lineStr.length() + outWordStr.length()) > MAX_LENGTH)
221     {
222         cout << lineStr << endl;
223         lineStr.clear();
224     }
225
226     lineStr = lineStr + outWordStr + " ";
227     outWordStr.clear();
228
```

```
229     cout << lineStr << endl;
230     lineStr.clear();
231 }
232
```