Justin Nguyen
JJ Javier

5/13/23

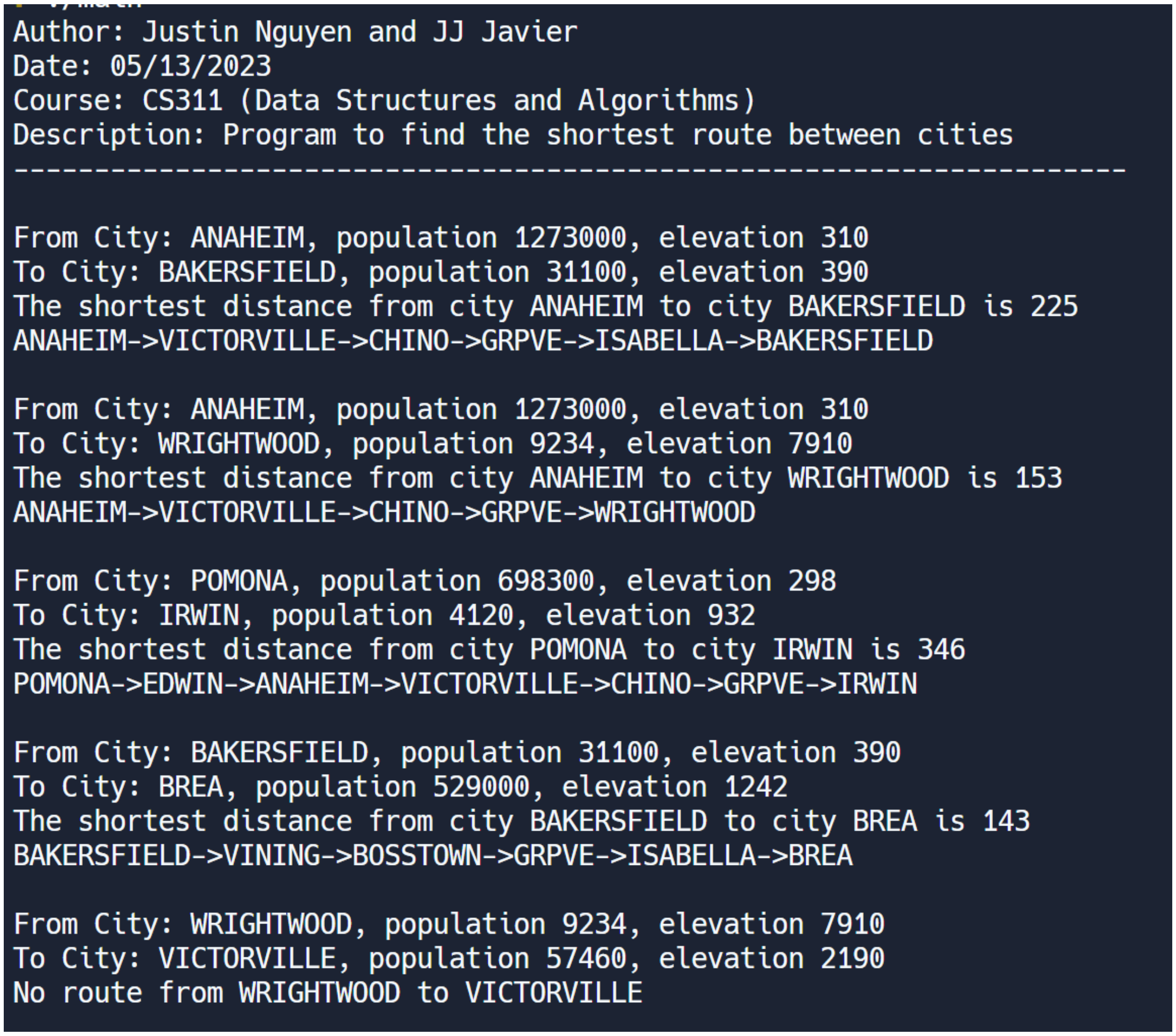
Group Assignment 8

For this assignment, based on a previous lecture on graphs, we originally attempted to use Dijkstra's algorithm to provide the shortest routes when going from one city to another city. We used replit in order to code together in real time and in doing so we started out by opening the "road.txt" file and using that file to find the shortest paths between different cities. After doing that we then later implemented the "city.txt" file. Doing this we turned City into an vector object oriented class in order to utilize the city data properly associated with their roads. From that we made a function that would read the "city.txt" file and put it into a City object for us to use.

After working on Dijkstra's algorithm for hours we later switched to Bellman-Ford's algorithm. We found this algorithm to be easier to use when dealing with the assignment prompt at hand. While Dijkstra's algorithm gave us trouble to figure out, switching the Bellman-Ford's algorithm allowed us to get our desired outputs in comparison to Dijkstra's. Based on the lecture, Bellman-Ford's algorithm is applicable here because on the first pass, we find the vertices that only have one edge, on the second pass, we use the d[u] values used to compute the correct d values for vertices whose shortest paths have two edges. After all the passes go through, all the d values should be correct.

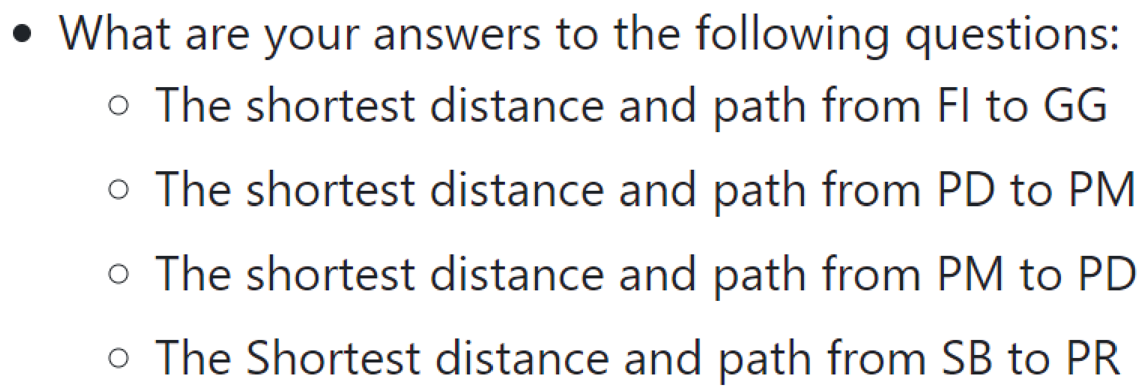
Problems we ran into was that dijkstra's algorithm wasn't working well with the directed graph we made for the assignment and the easy fix for us was simply just switching to use Bellman-Ford's algorithm which in turn gave us exactly what we needed for the desired outputs.

Our final outputs finally matched up with the “sample_results.txt” file with both the distances and the pathings being the shortest paths for each specific from city to city example.



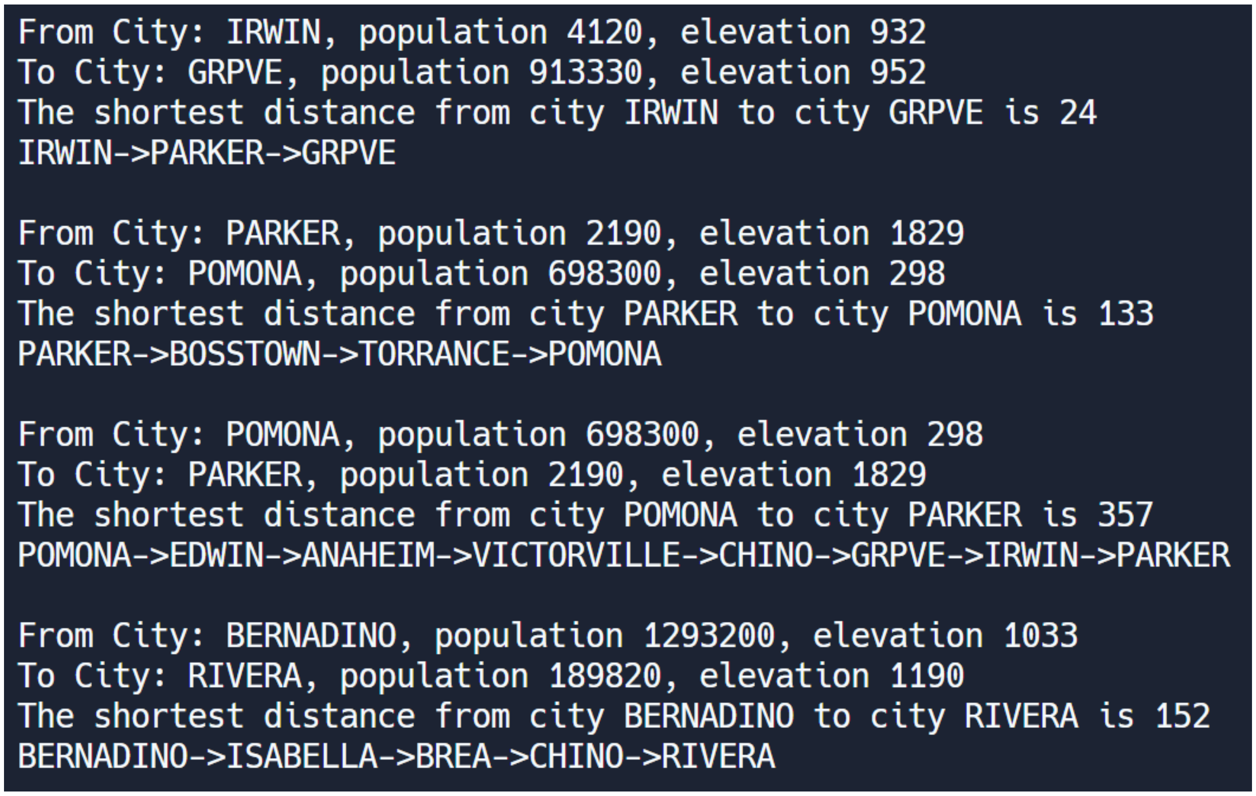
```
Author: Justin Nguyen and JJ Javier
Date: 05/13/2023
Course: CS311 (Data Structures and Algorithms)
Description: Program to find the shortest route between cities
--------------------------------------------------------------------

From City: ANAHEIM, population 1273000, elevation 310
To City: BAKERSFIELD, population 31100, elevation 390
The shortest distance from city ANAHEIM to city BAKERSFIELD is 225
ANAHEIM->VICTORVILLE->CHINO->GRPVE->ISABELLA->BAKERSFIELD

From City: ANAHEIM, population 1273000, elevation 310
To City: WRIGHTWOOD, population 9234, elevation 7910
The shortest distance from city ANAHEIM to city WRIGHTWOOD is 153
ANAHEIM->VICTORVILLE->CHINO->GRPVE->WRIGHTWOOD

From City: POMONA, population 698300, elevation 298
To City: IRWIN, population 4120, elevation 932
The shortest distance from city POMONA to city IRWIN is 346
POMONA->EDWIN->ANAHEIM->VICTORVILLE->CHINO->GRPVE->IRWIN

From City: BAKERSFIELD, population 31100, elevation 390
To City: BREA, population 529000, elevation 1242
The shortest distance from city BAKERSFIELD to city BREA is 143
BAKERSFIELD->VINING->BOSSTOWN->GRPVE->ISABELLA->BREA

From City: WRIGHTWOOD, population 9234, elevation 7910
To City: VICTORVILLE, population 57460, elevation 2190
No route from WRIGHTWOOD to VICTORVILLE
```

For the other results for,



- What are your answers to the following questions:
  - The shortest distance and path from FI to GG
  - The shortest distance and path from PD to PM
  - The shortest distance and path from PM to PD
  - The Shortest distance and path from SB to PR

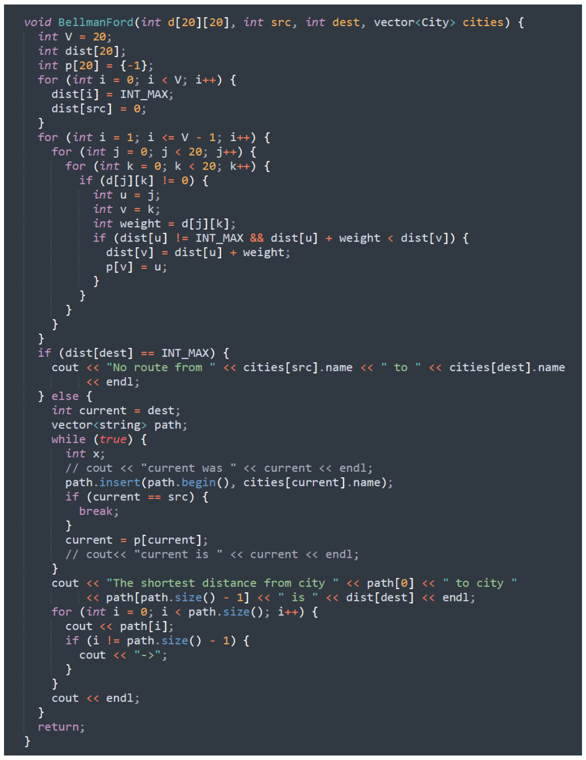
We get these results:



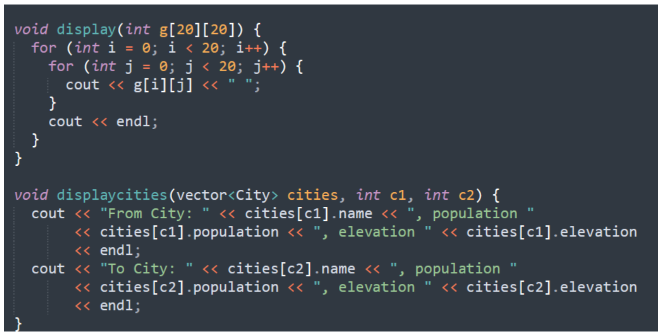
```
From City: IRWIN, population 4120, elevation 932
To City: GRPVE, population 913330, elevation 952
The shortest distance from city IRWIN to city GRPVE is 24
IRWIN->PARKER->GRPVE

From City: PARKER, population 2190, elevation 1829
To City: POMONA, population 698300, elevation 298
The shortest distance from city PARKER to city POMONA is 133
PARKER->BOSSTOWN->TORRANCE->POMONA

From City: POMONA, population 698300, elevation 298
To City: PARKER, population 2190, elevation 1829
The shortest distance from city POMONA to city PARKER is 357
POMONA->EDWIN->ANAHEIM->VICTORVILLE->CHINO->GRPVE->IRWIN->PARKER

From City: BERNADINO, population 1293200, elevation 1033
To City: RIVERA, population 189820, elevation 1190
The shortest distance from city BERNADINO to city RIVERA is 152
BERNADINO->ISABELLA->BREA->CHINO->RIVERA
```

Conclusion:
In conclusion, we ran into many problems when it mostly came into using Dijkstra's algorithm, but were able to overcome it with the implementation of a City class and the use of Bellman-Ford's algorithm. Though Bellman-Ford's algorithm has a longer runtime, it was the better choice for us to use and implement when trying to get the shortest paths from the "city.txt" and "road.txt" files. After making the City class, readCities function to put the data from the text file into the City class object, display function that shows graph, displayCities function that shows which 2 cities we are going from and to along with their population and elevation, and finally the Bellman-Ford function, we were successfully able to complete the assignment with the desired outputs.

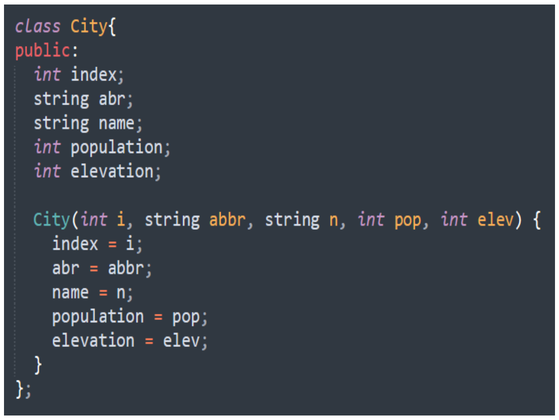
Appendix:



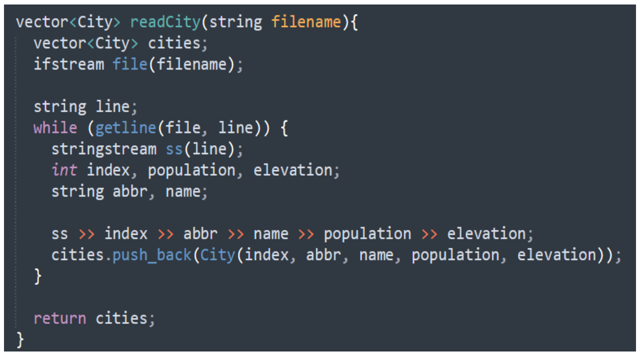
```cpp
void BellmanFord(int d[20][20], int src, int dest, vector<City> cities) {
  int V = 20;
  int dist[20];
  int p[20] = {-1};
  for (int i = 0; i < V; i++) {
    dist[i] = INT_MAX;
    dist[src] = 0;
  }
  for (int i = 1; i <= V - 1; i++) {
    for (int j = 0; j < 20; j++) {
      for (int k = 0; k < 20; k++) {
        if (d[j][k] != 0) {
          int u = j;
          int v = k;
          int weight = d[j][k];
          if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
            dist[v] = dist[u] + weight;
            p[v] = u;
          }
        }
      }
    }
  }
  if (dist[dest] == INT_MAX) {
    cout << "No route from " << cities[src].name << " to " << cities[dest].name
        << endl;
  } else {
    int current = dest;
    vector<string> path;
    while (true) {
      int x;
      // cout << "current was " << current << endl;
      path.insert(path.begin(), cities[current].name);
      if (current == src) {
        break;
      }
      current = p[current];
      // cout<< "current is " << current << endl;
    }
    cout << "The shortest distance from city " << path[0] << " to city "
        << path[path.size() - 1] << " is " << dist[dest] << endl;
    for (int i = 0; i < path.size(); i++) {
      cout << path[i];
      if (i != path.size() - 1) {
        cout << "->";
      }
    }
    cout << endl;
  }
  return;
}
```



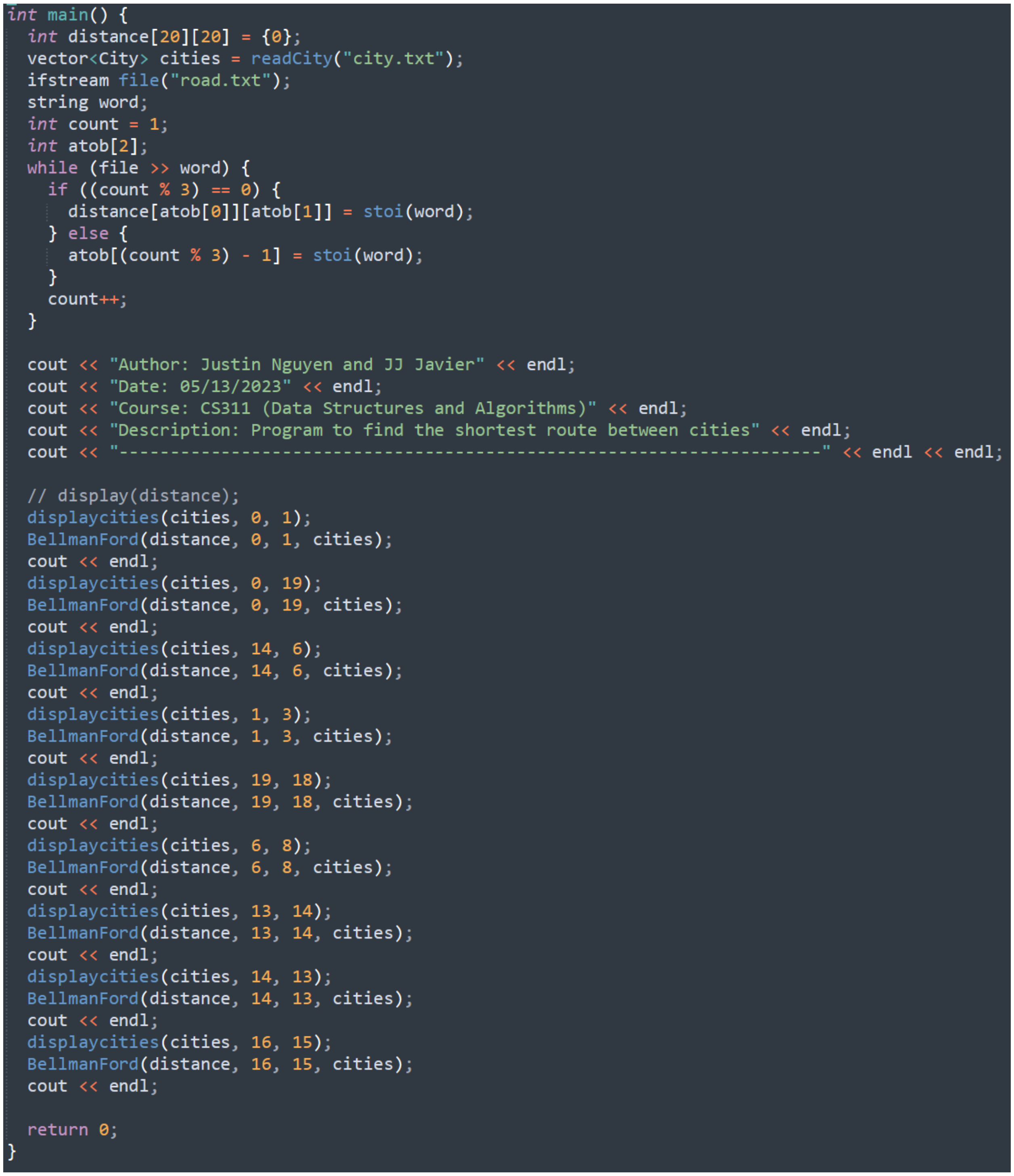
```cpp
void display(int g[20][20]) {
  for (int i = 0; i < 20; i++) {
    for (int j = 0; j < 20; j++) {
      cout << g[i][j] << " ";
    }
    cout << endl;
  }
}

void displaycities(vector<City> cities, int c1, int c2) {
  cout << "From City: " << cities[c1].name << ", population "
      << cities[c1].population << ", elevation " << cities[c1].elevation
      << endl;
  cout << "To City: " << cities[c2].name << ", population "
      << cities[c2].population << ", elevation " << cities[c2].elevation
      << endl;
}
```



```cpp
class City{
public:
  int index;
  string abr;
  string name;
  int population;
  int elevation;

  City(int i, string abbr, string n, int pop, int elev) {
    index = i;
    abr = abbr;
    name = n;
    population = pop;
    elevation = elev;
  }
};
```



```cpp
vector<City> readCity(string filename){
  vector<City> cities;
  ifstream file(filename);

  string line;
  while (getline(file, line)) {
    stringstream ss(line);
    int index, population, elevation;
    string abbr, name;

    ss >> index >> abbr >> name >> population >> elevation;
    cities.push_back(City(index, abbr, name, population, elevation));
  }

  return cities;
}
```


```cpp
int main() {
  int distance[20][20] = {0};
  vector<City> cities = readCity("city.txt");
  ifstream file("road.txt");
  string word;
  int count = 1;
  int atob[2];
  while (file >> word) {
    if ((count % 3) == 0) {
      distance[atob[0]][atob[1]] = stoi(word);
    } else {
      atob[(count % 3) - 1] = stoi(word);
    }
    count++;
  }

  cout << "Author: Justin Nguyen and JJ Javier" << endl;
  cout << "Date: 05/13/2023" << endl;
  cout << "Course: CS311 (Data Structures and Algorithms)" << endl;
  cout << "Description: Program to find the shortest route between cities" << endl;
  cout << "----------------------------------------------------------------------" << endl << endl;

  // display(distance);
  displaycities(cities, 0, 1);
  BellmanFord(distance, 0, 1, cities);
  cout << endl;
  displaycities(cities, 0, 19);
  BellmanFord(distance, 0, 19, cities);
  cout << endl;
  displaycities(cities, 14, 6);
  BellmanFord(distance, 14, 6, cities);
  cout << endl;
  displaycities(cities, 1, 3);
  BellmanFord(distance, 1, 3, cities);
  cout << endl;
  displaycities(cities, 19, 18);
  BellmanFord(distance, 19, 18, cities);
  cout << endl;
  displaycities(cities, 6, 8);
  BellmanFord(distance, 6, 8, cities);
  cout << endl;
  displaycities(cities, 13, 14);
  BellmanFord(distance, 13, 14, cities);
  cout << endl;
  displaycities(cities, 14, 13);
  BellmanFord(distance, 14, 13, cities);
  cout << endl;
  displaycities(cities, 16, 15);
  BellmanFord(distance, 16, 15, cities);
  cout << endl;

  return 0;
}
```