ECS 60 Programming Assignment #5, 50 points (my coding time 2.5 hours) Fall 2017

Due: **Thursday**, December 7th, 11:59pm.

Files NOT to handin to cs60 p5 are: EvacRunner.cpp, CPUTimer.h,

Minimum files to handin to cs60 p5 are: Makefile, evac.h, evac.cpp, authors.csv. Executable name: evac.out

You are to write an efficient C++ class, Evac, that will determine the routes taken to evacuate a group of cities that are geographically close to each other. Your class constructor will be passed an array of City. The City class and the Road class objects it contains are listed on the back of this handout. After main() destroys the City array, it calls Evac::evacuate, passing it an array of City that must be evacuated. The method fills an array or EvacRoutes with the movement of the people out of the evacuated cities into other cities. I have provided the driver program EvacRunner.cpp, and the necessary class stubs evac.h. You may add any classes and/or code you wish to evac.cpp, and evac.h. You may also use, and alter any Weiss files. CreateRoads.out was compiled from CreateRoads.cpp, and creates the evac files used for testing. All files mentioned, as well as my executable, evac.out, can be found in ~ssdavis/60/p5

Further specifications:

1. Command Line parameter is the name of the City file.
2. Evac Files
   - 2.1. Since EvacRunner.cpp handles all file input, you need not concerned with how to read them.
   - 2.2. File names are appended with three numbers: number of cities, number of bidirectional roads, and the seed used for the random number generator.
   - 2.3. The first line of the file contains the number of cities, number of roads, and the number of cities that will be evacuated.
   - 2.4. The second line of the file contains the IDs of the cities that must be evacuated. Their population will total around 10% of the total population. The cities are all close to each other geographically.
   - 2.5. Each of the remaining lines of the file contains information about one city including its ID, coordinates, population, and roads to adjacent cities.
     - 2.5.1. There is at most one bidirectional road between any pair of cities. Each direction of a road has the same peoplePerHour, but a different ID. So there are two times as many IDs as bidirectional roads.
3. Evacuation
   - 3.1. Your class must keep a simulated clock that is initialized to 1, and counts hours. Each Road has a variable indicating how many people it can transport in an hour. You will note that the EvacRoute class has a variable to hold the time of transport.
   - 3.2. Non-evacuated cities can only hold as many evacuees as their original population. For example, a City with a population of 4000 could support 4000 additional evacuees. You will find that your program will have to continue to transport many evacuees far from their original cities to find cities that can provide them beds. No city may ever have more evacuees in it than its own population. The number of evacuees in a city is checked at the end of each simulated hour.
   - 3.3. Only evacuated cities may have more people leave than have entered the city.
4. Measurements
   - 4.1. CPU time starts just before constructing your Evac object in main(), and ends after your evacuate() function returns. Thus, your destructors will not be called during CPU time.
     - 4.1.1. You may not have any global variables since they would be constructed before CPU time starts.
   - 4.2. Evac time is the number of simulated hours it took your class to completely empty the designated cities.
5. Grading
   - 5.1. The program will be tested using three 10000 city, 100,000 road files. The measurements will be the total of the three runs.
   - 5.2. If there are ANY error messages from EvacRunner, then the program will receive zero.
   - 5.3. If your code has no error messages, and takes less than 60 CPU time, then you will receive at least 20 points.
   - 5.4. CPU Time score = min(18, 15 * Sean's CPU / Your CPU)
   - 5.5. Simulated time score = min(18, 15 * Sean's Simulated time / Your simulated time)
6. Makefile. The Makefile provided contains the minimum needed to create evac.out.
   - 6.1. You may not use an optimization flags in your Makefile.
   - 6.2. All code must be C++ source code.
   - 6.3. To ensure that you handin all of the files necessary for your program, you should create a temp directory, and copy only *.cpp, *.h, and Makefile into it. Then try to make the program. Many students have forgotten to handin dsexceptions.h.

```
 class Road                          class City
 {                                   {
 public:                             public:
   int destinationCityID;              int ID;
   int peoplePerHour;                  int x;
   int ID;                             int y;
 }; // class Road                     int population;
                                     int evacuees;
 class EvacRoute                     Road *roads;
 {                                   int roadCount;
 public:
   int roadID;                       City();
   int time;                         ~City();
   int numPeople;                  }; // class City
   bool operator< (const
 EvacRoute &rhs) const;
 }; // EvacRoute;

int main(int argc, char* argv[])
{
  int numCities, numRoads, numEvacs, routeCount;
  ifstream inf(argv[1]);
  inf >> numCities >> numRoads >> numEvacs;
  City *cities = new City[numCities];
  int *evacIDs = new int[numEvacs];
  readCities(inf, cities, evacIDs, numCities, numEvacs);
  EvacRoute *evacRoutes = new EvacRoute[numCities * 1000];
  CPUTimer ct;
  ct.reset();
  Evac *evac = new Evac(cities, numCities, numRoads);
  delete [] cities;
  evac->evacuate(evacIDs, numEvacs, evacRoutes, routeCount);
  cout << "CPU Time: " << ct.cur_CPUTime() << endl;
  ifstream inf2(argv[1]);
  inf2 >> numCities >> numRoads >> numEvacs;
  cities = new City[numCities];
  Road2 *roads = new Road2[2 * numRoads];
  char *evacIDs2 = new char[numCities];
  readCities2(inf2, cities, evacIDs2, numCities, numEvacs, roads);
  checker(cities, evacIDs2, numCities, numEvacs, evacRoutes, routeCount, roads);
  return 0;
}

[ssdavis@lect1 p5]$ evac.out cities-10000-100000-4.txt
CPU Time: 6.79588
Evacuation hours: 165
[ssdavis@lect1 p5]$ evac.out cities-10000-100000-5.txt
CPU Time: 3.65918
Evacuation hours: 88
[ssdavis@lect1 p5]$ evac.out cities-10000-100000-6.txt
CPU Time: 6.76214
Evacuation hours: 72
[ssdavis@lect1 p5]$
```