ECS 60 Programming Assignment #5, 50 points (2.5 hours coding) Winter 2016

Due: Sunday, March 13th at 11:59pm to p5 directory
Primary class file name: maxfinder.cpp maxfinder.h Executable name: netdriver.out Makefile name: Makefile
Minimum files to submit: authors.csv, maxfinder.cpp, maxfinder.h, Makefile.

Candidate Bernie Sanders would like to know how many donations his current network can handle. Your MaxFinder class will accept a list of computers with their IP addresses. The constructor for MaxFinder also receives a number, numTerminals, indicating how many computers have direct connections with donors, and thus are the starting points. The first numTerminals computers in the list are the terminals. The last computer listed is the final destination for all donation information. The calcMaxFlow() function receives a list of edges which includes their capacities. No two computers will be connected by more than one edge. MaxFinder must determine the maximum number of donations that can get from all of the terminals to the final destination computer in one network cycle.

"Depth" is the length of the path from a terminal to the final destination computer. The depth difference of two connected computers never exceeds one, and could be zero. The expected length of the path from a terminal to the destination computer is (numComputers - numTerminals – 1) / numTerminals. Unlike other computers, the destination computer has at least as many inputs as there are terminals. For more information about the design of the network look at CreateNet.cpp.

You will find data files, CreateNet.out, CreateNet.cpp, NetDriver.h, NetDriver.cpp, maxfinder.cpp, maxfinder.h, a Makefile, and my executable in ~ssdavis/60/p5.

1. Grading
   - 1.1. Performance will be tested with three data files with 2000 to 5000 computers, 20 to 500 terminals, and two to three times as many edges as computers.
   - 1.2. I will copy NetDriver.h and NetDriver.cpp into directory, and then call make, so please make sure you handin all necessary files. Students often forget to handin dsexceptions.h, as well as other secondary Weiss files. "handin cs60 p5 authors.csv *.cpp *.h Makefile" would probably be wise after a successful clean remake.
   - 1.3. (25 points) Correctly determines the maximum number of donations. If the number is incorrect then the program will receive zero for the entire assignment.
   - 1.4. (25 points) CPU time: min (30, 25 * Sean's CPU Time / Your CPU Time)..
     - 1.4.1. CPU time may not exceed 60.
     - 1.4.2. Programs must be compiled without any optimization options. You may not use any precompiled code, including the STL and assembly.
     - 1.4.3. You may not have any large static, or global variables since they would be created before the CPU timer begins. Note that you may have constants.
2. Suggestions
   - 2.1. Keep things simple, and get things running first, and only then use gprof to learn where things are going slowly.
   - 2.2. Keep the address of computers and/or edges stored as a string during development to make identifying things easier during debugging. Once you are finished with debugging, you can easily eliminate them to pick up some space and thus time.
   - 2.3. You might think of adding the capability of printing each augmented path.
   - 2.4. Since there will be only two or three times as many edges, and a random number generator is involved in assignments, the number of adjacent vertices and back flows has an expected upper bound.
   - 2.5. Use Weiss code where possible.
   - 2.6. Remember to turn in dsexceptions.h if your program needs it!

```
struct Computer
{
  char address[16];
};

struct Edge
{
  char src[16];
  char dest[16];
  short capacity;
};

int main(int argc, char* argv[])
{
  int numComputers, numTerminals, numEdges;
  CPUTimer ct;

  ifstream inf(argv[1]);
  inf >> numComputers >> numTerminals >> numEdges;
  Computer *computers = new Computer[numComputers];
  Edge *edges = new Edge[numEdges];

  for(int i = 0; i < numComputers; i++)
    inf >> computers[i].address;

  for(int i = 0; i < numEdges; i++)
    inf >> edges[i].src >> edges[i].dest >> edges[i].capacity;

  ct.reset();
  MaxFinder *maxFinder = new MaxFinder(computers, numComputers,numTerminals);
  delete [] computers;
  int flow = maxFinder->calcMaxFlow(edges, numEdges);
  cout << "CPU: " << ct.cur_CPUTime() << " Flow: " << flow << endl;
  return 0;
} // main()

[ssdavis@lect1 p5]$ cat net-7-2-7-3.txt
7 2 7
43.209.93.90
84.215.89.192
176.24.154.60
155.201.73.128
121.208.145.240
120.126.212.40
139.78.152.11
43.209.93.90 121.208.145.240 6
121.208.145.240 176.24.154.60 20
176.24.154.60 155.201.73.128 20
155.201.73.128 139.78.152.11 20
84.215.89.192 120.126.212.40 23
120.126.212.40 139.78.152.11 23
176.24.154.60 120.126.212.40 20
[ssdavis@lect1 p5]$ netdriver.out net-7-2-7-3.txt
CPU: 9.4e-05 Flow: 29
[ssdavis@lect1 p5]$
```