**ECS 60 Programming Assignment #5, 50 points (2 hours coding) Fall 2016**

Due: **Thursday**, December 1st at 11:59pm to p5 directory
Minimum files to submit: blood.cpp, blood.h, authors.csv Executable name: blood.out Makefile name: Makefile

You are to write a program that determines the minimum blood flow pattern for a brain. You program will be given a list of brain cells and how blood vessels connect them, starting from the carotid artery, and ending with the jugular vein. Your program will determine how much blood should flow between connected brain cells during a series of pulses until all the brain cells have received at least one full blood cell.

You are to handin all files upon which your program is dependent, except bloodRunner.cpp, bloodRunner.h, and CPUTimer.h. The grading script will copy those three files into your directory. You must handin a Makefile! You will find those three files, brain files, and my blood.out. in ~ssdavis/60/p5.

Here are the specifications:

1. CreateFile.cpp is available and worth perusing.
   - 1.1. The name of a data file reflects parameters used to create it. For example, brain-25-100-5.txt has 25 brain cells, 100 blood vessels, and was created using a seed of 5 for the random number generator.
   - 1.2. The first line of the file contains the number of brain cells followed by the number of blood vessels, and the length of the shortest path between the first and last cells.
   - 1.3. Each succeeding lines contains information about one blood vessel: <ID> <upstream brain cell ID> <downstream brain cell ID> <carrying capacity>
2. Brain Cells
   - 2.1. Each brain cell has a unique ID. They are assigned their IDs randomly, though the IDs are contiguous. However, the first cell encountered by the carotid artery is ID 0, and the last cell before the jugular vein will have the largest ID.
   - 2.2. Your program should indicate when every brain cell has received a full blood cell.
   - 2.3. Each brain cell now empties exactly one full blood cell during the life of the program. Thus, if cell B is downstream from cell A, and cell A has never received a full blood cell before, then cell A must receive at least two full blood cells during a pulse so that at least one full blood cell will make it to cell B. Note that blood cells are not eaten by the brain cells, and must continue along the blood vessels once emptied.
3. Blood Vessel
   - 3.1. Each blood vessel has a unique ID. They are assigned their IDs randomly, though the IDs are contiguous, and start at zero.
   - 3.2. Each blood vessel connects two brain cells. The blood flows from the first cell specified to the second cell.
   - 3.3. Up to five blood vessels can connect to a single brain cell, except the last may have any number of vessels leading into it.
   - 3.4. Each blood vessel has a blood cell carrying capacity.
   - 3.5. While in a real brain it takes time to transport blood cells, we will assume that they are transported instantly through the whole brain. (You really don't want to simulate time do you?).
     - 3.5.1. Oddly, the system may be thought of as empty at the end of each pulse.
   - 3.6. There are no cycles in the blood vessels for this could cause recycling of empty blood cells.
   - 3.7. The total number of blood cells arriving at a brain cell must equal the total number leaving. When a brain cell empties a blood cell, the number of full cells leaving the brain cell would be one less than entering, and the number of empty cells would be one more than entering. The first and last cells do not obey this conservation of blood rule. The first cell can provide an infinite number of full blood cells, and the last cell can absorb an infinite number of blood cells.
   - 3.8. Each pulse, your class is provided two arrays of ints with one position for each blood vessel. One array is for the flow of full blood cells, and the other is for the flow of empty blood cells in the blood vessels. For each pulse, the class fills each position with the number of blood cells of each type that the corresponding vessel must carry.
4. Grading
   - 4.1. Performance will be tested with three data files, each with 5000 brain cells, and 10000 vessels . Each of these values is the maximum possible for the program.
   - 4.2. (10 points) Correctly feed all cells. This is indicated by having no messages from checkFlows(). If a program does not correctly process the instructions, then it will receive zero for the entire assignment.
   - 4.3. (20 points) CPU time: min (23, 20 * Sean's CPU Time / Your CPU Time);

4.3.1. The program terminates when all brain cells have been fed at least one full blood cell, or the number of pulses reaches **10000**.
4.3.2. CPU time may not exceed 60.
4.3.3. Programs must be compiled without any optimization options. You may not use any precompiled code, including the STL and assembly.
4.3.4. You may not have any static, or global variables since they would be created before the CPU timer begins.
4.4. (20 points) Simulated pulses to feed all brain cells
4.4.1. min(23, 20 * (Sean's simulated pulses) / (Your simulated pulses)

5. Suggestions
5.1. Keep things simple, and get things running first, and only then use gprof to learn where things are going slowly.
5.2. Use Weiss code as a starting point where possible.
5.3. Remember to turn in dsexceptions.h if your program needs it!

```
int main(int argc, char *argv[])
{
  int vesselCount, cellCount, depth, *emptyFlows, *fullFlows, pulses = 0, totalFed = 0,
theirTotalFed;
  Vessel *vessels, *vessels2;
  Cell *cells;
  CPUTimer ct;

  if(argc != 2)
  {
    cout << "Usage: blood.out filename\n";
    return 1;
  } // if wrong number of arguments

  readFile(argv[1], &vessels, &vessels2, &vesselCount, &cellCount, &depth);
  cells = new Cell[cellCount];
  emptyFlows = new int[vesselCount];
  fullFlows = new int[vesselCount];
  ct.reset();
  Blood *blood = new Blood(vessels2, vesselCount, cellCount, depth);
  delete [] vessels2;

  do{
    theirTotalFed = blood->calcFlows(fullFlows, emptyFlows);
    checkFlows(vessels, fullFlows, emptyFlows, cells, vesselCount, cellCount,
      pulses, &totalFed);
    if(theirTotalFed != totalFed)
      cout << "At pulse #" << pulses << " your number fed, " << theirTotalFed
        << ", does not match ours, " << totalFed << endl;
  } while(++pulses < 10000  && totalFed < cellCount);

  cout << "Time: " << ct.cur_CPUTime() << " Ticks: " << ct.cur_CPUTicks()
    << " Pulses: " << pulses << endl;

  for(int i = 0; i < cellCount; i++)
    if(!cells[i].fed)
      cout << "Cell #" << i << " has not been fed.\n";

  return 0;
}  // main()

[ssdavis@lect1 p5]$ blood.out brain-5000-10000-20.txt
Time: 0.764036 Pulses: 235
[ssdavis@lect1 p5]$ bloodGold.out brain-5000-10000-20.txt
Time: 0.32788 Pulses: 153
[ssdavis@lect1 p5]$
```