ECS 36C **Programming Assignment #4 (50 points, 50 minutes)** **Spring 2021**

Due: May 19th, 11:59pm to p4 directory
Primary source file name: spreader.cpp, spreader.h Executable name: spreader.out Makefile name: Makefile
Minimum handin command: handin cs36c p4 spreader.cpp spreader.h Makefile authors.csv

You are to write a Spreader class that will determine how many days it would take for Covid to infect an entire population when beginning with a few starting individuals. The constructor of your class will receive an array of the people with information about who they regular visit, and how often they visit them. The name of a covid file will be passed as the first command line parameter to the program. Though a second commandline parameter may be passed for use in debugging, it can be safely ignored by your program.

For our program will assume that during a 3 day incubation period a newly infected person is not contagious. We also assume there is a uniform periodicity to people visiting other people. For example, person #374 will always visit person #1043 every 6 days. The range of the periods is random, and from 1 to 7 days. A person actually meets a group of people. The groups range in size from1 to 10. The number of people who start infected on day 0 can range from 1 to 10.

The createPopulation.out program, compiled from createPopulation.cpp, creates the covid data files.

You will find my executable, spreaderRunner.h, spreaderRunner.cpp, a barebones spreader.cpp, a barebones spreader.h, a barebones Makefile, createPopulation.cpp, createPopulation.out, and some covid data files in ~ssdavis/36c/p4. Remember to hand in ALL the files that your program depends upon, except CPUTimer.h, spreaderRunner.h, and spreaderRunner.cpp. These last three files will be copied into your directory before make is called.

Covid data file format:

1. Since spreaderRunner.cpp reads and processes these files, this information is just to help you understand the files. The Spreader class will never deal with the files directly.
2. Name format: covid-<population>-<number of starting people>-<seed for random number generator>.csv.
3. First line format: <population>,<number of starting people>, <first starter ID>, <second starter ID> …
4. All succeeding lines have the format:
   <Person ID#>, <# people visited>,<first person visited ID#>,<frequency of visits to first person>,<second person visited ID#>,<frequency of visits to second person> …

Grading:

The program will be run with three 2,000,000 person files.

1. Proper operation with correct number of days until full infection = 35 points
2. Time score is only possible if there are no errors. It is possible to earn five points extra credit.
   Time Score = min (20, 15 * Sean's Total CPU Time / Your Total CPU Time).
3. Programs must be compiled without any optimization options. You may not use any precompiled code, including the STL and assembly. Other than Weiss files, you must have written all of the code you submit.
4. You may not have any static, or global variables larger than 1000 bytes, since they would be created before the CPU timer begins.
5. If your CPU time exceeds 30 seconds for a 2,000,000 person file, then please send me an e-mail when you submit your program so I can run it separately. If you do not send me such a warning, I will deduct 5 points from your score.
6. Miscellaneous:
   6.1. Brainstorm a lot before starting to code. Think about what you need to know about each Person, and add variables to your own Person2 class that is based on the Person class.
   6.2. I may help with debugging, but not design.
   6.3. Keep things simple, and get things running first, and only then use gprof to learn where things are going slowly.
   6.4. While I did use Weiss ADT(s), I eliminated the use of his Vector class to avoid the problem we found in CursorList.
   6.5. memcpy of <cstring> is the fastest way to copy a fixed amount of data.
   6.6. Remember to turn in dsexceptions.h if your program needs it! I suggest you create a directory, copy all of the files you think you need into it, and then try compiling before you use handin.

```
pc10:~/36c/p4$ cat spreaderRunner.h
#ifndef SPREADERRUNNER_H
#define SPREADERRUNNER_H

#define INCUBATION_TIME 3

class Visitor
{
public:
  int ID;
  int period;
};

class Person
{
public:
  int ID;
  int peopleMet;
  Visitor *visitors;
};

#endif /* SPREADERRUNNER_H */
pc10:~/36c/p4$

int main(int argc, char** argv)
{
  Person *people;
  int population, *starterIDs, starterCount;
  people = readFile(argv[1], &population, &starterIDs, &starterCount);
  CPUTimer ct;
  Spreader *spreader = new Spreader(people, population);

  for(int i = 0; i < population; i++)
   delete[] people[i].visitors;

  delete[] people;
  int days = spreader->simulate(starterIDs, starterCount, argc);
  cout << "CPU Time: " << ct.cur_CPUTime() << ' '
     << "Days for complete infection: " << days << endl;
  return 0;
} // main()
```

```
pc10:~/36c/p4$ createPopulation.out
Population>> 20
Number of starters (1 - 10)>> 3
Seed>> 5
pc10:~/36c/p4$
pc10:~/36c/p4$ cat covid-20-3-5.csv
20,3,13,5,15
0,5,9,7,10,7,14,7,16,7,18,7
1,1,19,1
2,6,3,6,4,6,5,6,6,6,8,6,12,6
3,6,2,6,4,6,5,6,6,6,8,6,12,6
4,10,2,6,3,6,5,6,6,6,8,6,12,6,7,4,11,4,18,4,19,4
5,6,2,6,3,6,4,6,6,6,8,6,12,6
6,6,2,6,3,6,4,6,5,6,8,6,12,6
7,7,13,5,15,5,17,5,4,4,11,4,18,4,19,4
8,6,2,6,3,6,4,6,5,6,6,6,12,6
9,5,0,7,10,7,14,7,16,7,18,7
10,5,0,7,9,7,14,7,16,7,18,7
11,4,4,4,7,4,18,4,19,4
12,6,2,6,3,6,4,6,5,6,6,6,8,6
13,3,7,5,15,5,17,5
14,5,0,7,9,7,10,7,16,7,18,7
15,3,7,5,13,5,17,5
16,5,0,7,9,7,10,7,14,7,18,7
17,3,7,5,13,5,15,5
18,9,0,7,9,7,10,7,14,7,16,7,4,4,7,4,11,4,19,4
19,5,1,1,4,4,7,4,11,4,18,4
pc10:~/36c/p4$
pc10:~/36c/p4$ spreader.out covid-20-3-5.csv
CPU Time: 3.4e-05 Days for complete infection: 14
pc10:~/36c/p4$
For debugging, when there is an argc > 2, my Spreader::simulate() prints out
useful information as the virus proceeds through the population.  Here is a run
with a 1 as the second command line parameter.
pc10:~/36c/p4$ spreader.out covid-20-3-5.csv 1
Infected count: 1 day: 0 ID: 13
Infected count: 2 day: 0 ID: 15
Infected count: 3 day: 0 ID: 5
Infected count: 4 day: 5 ID: 7
Infected count: 5 day: 5 ID: 17
Infected count: 6 day: 6 ID: 12
Infected count: 7 day: 6 ID: 3
Infected count: 8 day: 6 ID: 4
Infected count: 9 day: 6 ID: 8
Infected count: 10 day: 6 ID: 6
Infected count: 11 day: 6 ID: 2
Infected count: 12 day: 8 ID: 19
Infected count: 13 day: 8 ID: 18
Infected count: 14 day: 8 ID: 11
Infected count: 15 day: 11 ID: 1
Infected count: 16 day: 14 ID: 14
Infected count: 17 day: 14 ID: 10
Infected count: 18 day: 14 ID: 9
Infected count: 19 day: 14 ID: 0
Infected count: 20 day: 14 ID: 16
CPU Time: 0.000211 Days for complete infection: 14
pc10:~/36c/p4$
```