ECS 60 **Programming Assignment #5, 50 points** **Winter 2015**

Due Sunday, March 15th , 11:59pm. Name of executable is p5.out

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

Minimum files to handin to cs60 p5 are: Makefile, scheduler.cpp, scheduler.h

You are to write an efficient program that will determine the number of people to assign to each job in a large project. I have provided the driver program ProjectDriver.cpp, and the necessary class stubs in scheduler.h, and scheduler.cpp. You may add any classes and/or code you wish to scheduler.cpp, and scheduler.h. You may also use, and alter any other files you wish, including the Weiss files. WorkCreate.out was compiled from WorkCreate.cpp, and creates files used for testing. All files mentioned, as well as my executable, ProjectDriver.out, can be found in ~ssdavis/60/p5.

Further specifications:

1. Command Line parameters: The first is name of the operations file, and the second is the number of people to use for the project. Any output other than the final CPU Time line and the finish time of the project will be considered an error, and earn a zero.
2. Scheduler class
   - 2.1. The constructor for the Scheduler class accepts an array of Job that contains all of the information about the project. When Scheduler::run() is called, your program will decide which person to assign to which job. The scheduler indicates its choices by placing the person's ID in the peopleIDs array of the job, and setting the numPeopleUsed of that job accordingly. Before returning from run(), the scheduler must set each job's startTime.
3. Error Checking. The program checks your jobs array based on the following rules.
   - 3.1. The time taken to complete a job is
     int (ceil(jobs[ID].length / double(jobs[ID].numPeopleUsed)));
   - 3.2. Each person is assigned to a job for its duration, and may not work on another job during that time.
   - 3.3. No job may start before all of its ancestors are finished.
   - 3.4. All jobs in the project must be completed.
4. Project Files
   - 4.1. These are created by WorkCreate.out. The names reflect the number of jobs, maximum number of workers per job, and the seed used to initialize the random number generator.
   - 4.2. The ProjectDriver completely handles these file so you are not responsible for parsing them, however you may need to understand them for debugging purposes.
   - 4.3. Each line starts with a job ID, then the number of person-days to complete the job (never more than 50), then the number of jobs directly dependent on the completion of the job, and finally a list of all those jobs.
5. Measurements
   - 5.1. CPU time starts just before constructing your Scheduler object in main(), and ends when it returns from its run() call. Thus, your destructors will not be called during CPU time.
     - 5.1.1. You may not have any global variables since they would be constructed before CPU time starts.
     - 5.1.2. Your program may not be compiled with any optimization flags.
     - 5.1.3. You may not use any precompiled code, including the STL and assembly.
   - 5.2. The finish time is the latest finish time among all of the nodes.
   - 5.3. The program will be tested using three 5000 job files with a maximum number of workers per job of 25. One will have 500 people, another 1000 people, and the last will have 2000. The measurements will be the total of the three runs.
     - 5.3.1. You are guaranteed that there are enough people to cover all paths in the file. If the number of people is too small not all jobs can be easily serviced, which would make the programming needlessly complex.
   - 5.4. If there are ANY error messages when run, then the program will receive zero.
   - 5.5. If your program successfully navigates through all three files in less than 60 CPU units you will earn 25 points.
   - 5.6. CPU Time score = min(12, 10 * Sean's CPU / Your CPU)
   - 5.7. Finish time = min(18, 15 * Sean's Finish time / Your Finish Time)
6. Makefile. The Makefile provided contains the minimum needed to create p5.out.
   - 6.1. 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.
7. Suggestions.
   - 7.1. Start early, and get something working without errors.

7.2. Don't fuss about anything until you have something working. Too many students fail to have a correct program by the deadline because they spend too much time tweaking early on.
   7.2.1. You will learn a lot just getting it running. Then you can use this knowledge to improve your code.
7.3. To debug, add if-statements that are only true during the state in which you are interested. For example, if I had error when I was working with job #134, then I would add to my Scheduler::run() code:

```
if(ID == 134)
   ID = 134;
```

I then can set a breakpoint at the "ID = 134;" line, and step through the code to see what was going on.

7.4. Leave lots of time for testing and debugging. Test with all of the available files. The large files are almost impossible to debug. If you can make smaller files that duplicate the problem. You will find ancestors.out helpful. It takes a file name as a parameter and then prints out all of the ancestors of a given job.

```
typedef struct
{
  short length;
  short numDependencies;
  int startTime;
  int finishTime;
  short numPeopleUsed;
  int dependencies[50];
  short peopleIDs[50];
} Job;
```

```
[ssdavis@lect1 p5]$ cat Jobs-10-3-1.csv
0,37,1,6
1,24,1,9
2,8,0
3,46,0
4,32,2,0,3
5,24,2,0,2
6,46,0
7,30,0
8,21,3,0,7,2
9,35,3,8,5,4
[ssdavis@lect1 p5]$
```

```
[ssdavis@lect1 p5]$ ancestors.out Jobs-10-3-1.csv
Job # (-1 = done): 0
Job #    0 Len:  37    4    5    8
Job #    4 Len:  32    9
Job #    9 Len:  35    1
Job #    1 Len:  24
Job #    5 Len:  24    9
Job #    8 Len:  21    9

Job # (-1 = done): -1
[ssdavis@lect1 p5]$
```

```
[ssdavis@lect1 p5]$ p5.out Jobs-10-3-1.csv 3
CPU time: 0
Job ID# 5 has no people assigned to it.
Job ID# 2 has no people assigned to it.
Finish time: 3200020
[ssdavis@lect1 p5]$ p5.out Jobs-10-3-1.csv 4
CPU time: 0
Finish time: 122
[ssdavis@lect1 p5]
[ssdavis@lect1 p5]$ p5.out Jobs-5000-25-2.csv 500
CPU time: 1.53845
Finish time: 390
[ssdavis@lect1 p5]$ p5.out Jobs-5000-25-2.csv 1000
CPU time: 3.07761
Finish time: 191
[ssdavis@lect1 p5]$ p5.out Jobs-5000-25-2.csv 2000
CPU time: 6.16854
Finish time: 117
[ssdavis@lect1 p5]$
```