# COP 4600 – Programming Assignment #1 – Summer 2014

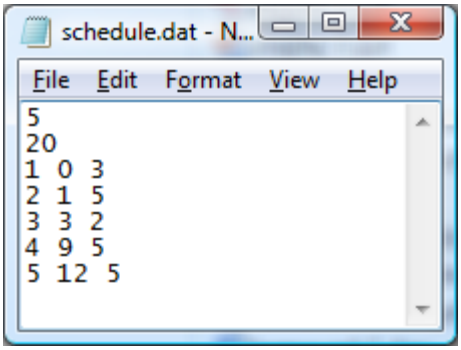
**Title:** "Program Assignment 1: Uniprocessor Scheduling Protocols"
**Points:** 100 points
**Due Date: Thursday July 3rd by 11:59 pm (WebCourses time)**

**Objectives:** To develop a simulation program for various uniprocessor scheduling protocols. This will help you to better understand the job of the dispatcher and how it operates.

**Description:** Develop a program using C, C++, or Java (other languages that I know would also be ok, but clear it with me first) that simulates various uniprocessor scheduling protocols and produces a Gantt chart of their execution profiles as well as producing the average waiting time, average turnaround time, and the normalized turnaround time for each process in the schedule of ready processes. The technique for producing the results is the same that you utilized in problem #3 in the first homework assignment. Implement the following scheduling protocols: FCFS (First Come – First Served), RR (Round-Robin) with time quantums of 1 and 3, and SPN (Shortest Process Next). Assume that the input to the program is in the form of a file (named "`schedule.dat`") containing only integer values with the following format:

line 1: COUNT, indicating the number of processes in the ready queue.
line 2: TIME, indicating the total processing time required by all processes in the ready queue.
x lines: PROCESS_ID ARRIVAL_TIME PROCESSING_TIME (where x = COUNT)

Using the set of processes from homework #1, problem #3 as an example, the input file representing these processes would look like the following (available on the course website):




For a given input file, your program should produce, as output, the Gantt chart and statistics mentioned above for each of the four protocols. An example of a possible output form is shown below, I used 3 different symbols for the possible process states: W = waiting, E = executing (i.e., process holds the CPU), T = terminated (process has completed). An empty cell in the matrix simply means that the process has not yet entered the ready queue.

**FCFS**

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | E | E | E | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T | T |
| 2 | | W | W | E | E | E | E | E | T | T | T | T | T | T | T | T | T | T | T | T |
| 3 | | | | W | W | W | W | W | E | E | T | T | T | T | T | T | T | T | T | T |
| 4 | | | | | | | | | | W | E | E | E | E | E | T | T | T | T | T |
| 5 | | | | | | | | | | | | | W | W | W | E | E | E | E | E |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |

| | wait | turnaround | normalized turnaround |
|---|---|---|---|
| 1 | 0 | 3 | 1 |
| 2 | 2 | 7 | 1.4 |
| 3 | 5 | 7 | 3.5 |
| 4 | 1 | 6 | 1.2 |
| 5 | 3 | 8 | 1.6 |
| AVG | 2.2 | 6.2 | 1.74 |

**References:**
Notes: Lecture Notes for Uniprocessor Scheduling.

**Restrictions:**
Your source file shall begin with comments containing the following information:

```
/* Name:
   Course: COP 4600 Summer 2014
   Assignment title: Program #1 – Uni-processor Scheduling Simulation
   Date: July 3, 2014
*/
```

**Input Specification:** The file "`schedule.dat`" as described above (see example below as well).

**Output Specification:** Output should be four different Gantt charts and sets of statistics corresponding to the four scheduling protocols (FCFS, RR with quantum = 1, RR with quantum = 3, and SPN).

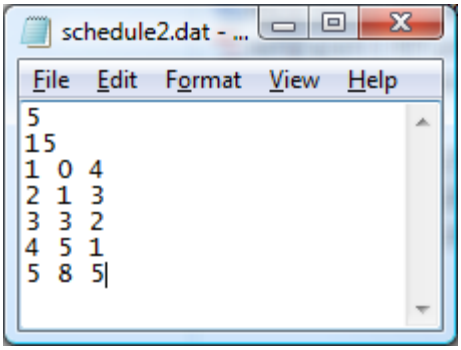
**Deliverables:**

(1) Submit a working copy of your source code via WebCourses no later than 11:59pm Thursday July 3rd .

(2) Screen shots illustrating your program operating on the sample `schedule.dat` input file shown above.

**Additional Information:**

Shown below is another sample input file




**Hints:**

Use an array to store the process states, a 2d one works fine.

Use a separate function/method for each scheduling protocol and have the driver function/method simply run through each of the protocols separately, rebuilding the Gantt chart each time (i.e., refilling the array values).

Make a parameter to the RR function/method that sets the time quantum to be used so that the function/method works for all time quanta.

We'll assume that the total processing time required by all processes (line 2 in the input file) will be small enough so that your entire Gantt chart will fit on one line and you won't need to consider breaking up the table into pieces. The formatting that I used means that this upper limit is about 25. You might be able to get more if you reduced the spacing and the grid lines, but we won't worry about that sort of detail.