COMP SCI 3004/7064 - Operating Systems Assignment 1

DUE: 11:30pm, 10th Sept, 2018

Important Notes

- Handins:
  - The deadline for submission of your assignment is **11:30pm Monday the 10th of Sept, 2018**.
  - **For undergraduate students**, you may do this assignment as a team of two students and you must hand in one submission per team.
  - **For postgraduate students**, you have to do this assignment individually and make individual submissions.
  - All implementations have to be done in **C**.
  - You need to submit your source code using the web submission system. You should attach you and your partner's name and student number in your submission.
  - Late submissions will attract a penalty: the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.
- Marking scheme:
  - 16 marks for auto-marked testing on 4 standard tests (4 marks per test).
  - 4 marks for the success of compiling and code structure.
  - **Note**: If it is found your code did not implement the algorithm described below, you will receive zero mark regardless of the correctness of test output.

If you have any questions, please send them to the student discussion forum. This way you can all help each other and everyone gets to see the answers.

The assignment

The aim of this assignment is to improve your learning experience in the process scheduling algorithms. You are required to design an online service system for a large ITS enterprise to handle customer requests. In the system, all the customers are grouped into six priority classes, ranged from 1 to 6, according to their membership status. A larger priority number indicates a higher priority. To maximise the system service performance, you are required to implement a scheduling algorithm using multi-level queue strategy with three queues: : a high priority Queue 1 (customers with priority number 6 and 5), a medium priority Queue 2 (priority number 4 and 3) and a low priority Queue 3 (priority number 2 and 1). Queue 1 has absolute priority over Queue 2 and Queue 3, and Queue 2 has absolute priority over Queue 3. In other words, customer requests in Queue 2 will be processed only if there is no customer in Queue 1, and customer requests in Queue 3 will be processed only if no customer is in Queue 1 and Queue 2. There are two thresholds (=5, 3) that are used to determine whether a customer should remain in Queue 1 (priority ≥ 5), in Queue 2 ($3 \geq$ priority < 5) and Queue 3 (priority < 3), respectively. Detailed actions in these three queues are listed below:

Queue 1: This is the high priority queue. Customers in this queue are treated in the way of combined *Highest Priority First* (HPF) and *Round Robin.* That is, select the highest priority customer (customers with the same priority are selected in their arrival order), and process this customer's request for a time quantum of 5 time units non-preemptively, then move this customer to the end of a sub-queue (in same priority) of Queue 1. The priority of a customer in this queue is decreased by one every 5 runs of this customer, i.e. once having been serviced 25 time units overall in Queue 1.

Queue 2 and Queue 3: These are the medium/low priority queue. Customers in these queues are handled in *Round Robin.* That is, select the customer with *First Come First Serve*, and process this customer's request for a time quantum of 10 time units for Queue 2 and 20 time units for Queue 3 *preemptively* as described below, then move this customer to the end of its queue. If the priority of a customer in these queues reaches the threshold (5 in Queue 2 and 3 in Queue 3) by the following Aging mechanism, that customer is promoted from Queue 2 to Queue 1 (or from Queue 3 to Queue 2). There's one more thing need to be noted: The priority of a customer in Queue 2 is decreased by one every 2 runs of this customer, i.e. once having been serviced 20 time units overall.

Preemption: Once a running customer in Queue 2 (or Queue 3) is interrupted by a new arrival customer from Queue 1 (or from Queue 2), this customer will leave its time quantum immediately and moved to the end of its queue.

Aging: Because Queue 1 has priority over Queue 2 and Queue 3, and the HPF strategy is applied, starvation may occur. That is, some customers in Queue 2 and Queue 3 may never get to run because there is always a customer with higher priority in Queue 1. To solve the starvation issue, you must implement a mechanism which ages each customer. This need not be done every time a customer run as this will slow the system down, but say once after every 7 customer runs. That is, if a customer has waited 7 runs of other customers since its last run, its priority number will be increased by one. In this way, the priority of each customer in Queue 2 and Queue 3 increases gradually in proportion to the waiting time since the last run.

Note: If at time t, there are three customers with the same priority: a new arrival customer A to Queue 1, a customer B moved to the end of Queue 1, a promoted customer C from Queue 2 to Queue 1, their order will be $A \rightarrow B \rightarrow C$. Same rule applies to Queue 2 and 3 regardless of the priority.

Test

Input

Each customer is identified by a line in the input file. The line describes the customer ID, arrival time, priority, age and the total time units required. For example s1 3 1 0 50 describes a customer s1 who arrived at time 3 with priority 1 and age 0, and requires 50 time units.

Output

The output provides information of each customer execution. The line starts with the customer ID, priority, arrival and termination times, ready time (the first time the system processes its request) and durations of running and waiting.

Web-submission instructions

- First, type the following command, all on one line (replacing xxxxxxx with your student ID):

```
svn mkdir --parents -m "OS"
https://version-control.adelaide.edu.au/svn/axxxxxxx/2018/s2/os/assignment1
```

- Then, check out this directory and add your files:

```
svn co https://version-control.adelaide.edu.au/svn/axxxxxxx/2018/s2/os/assignment1
cd assignment1
svn add OnlineService.c
svn commit -m "assignment1 solution"
```

- Next, go to the web submission system at:
  `https://cs.adelaide.edu.au/services/websubmission/`
  Navigate to 2018, Semester 2, Operating Systems, Assignment 1. Then, click Tab "Make Submission" for this assignment and indicate that you agree to the declaration. The automark script will then check whether your code compiles. You can make as many resubmissions as you like. If your final solution does not compile you won't get any marks for this solution.

- We will test your codes by the following Linux commands (read input as an argument to the main function, print the result to the screen as the output):

```
gcc -std=c11 OnlineService.c -o run
./run input.txt > output.txt
```

- Please avoid hard-coding when reading input file.