COMP SCI 3004/7064 - Operating Systems Assignment 1

DUE: 23:30pm, 16th Sept, 2019

Important Notes

- Handins:
  - The deadline for submission of your assignment is **23:30pm, 16th Sept, 2019**.
  - **For undergraduate students**, you may do this assignment as a team of two students and 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:
  - 20 marks for 10 randomly generated tests (2 marks per test).

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 ticketing system for Coopers Stadium. Specifically, there are two seat sections: red section for public booking via this system; non-red section for private booking via hot line. Figure 1 shows the distribution of the seat sections.

In the system, all the customers are grouped into seven priority classes(numbers), ranged from 1 to 7, according to their loyalty (accumulated points) to this ticketing system. A smaller priority number indicates a higher priority. All ticketing processes (purchase/cancellation) generated by a customer are assigned with the same priority number of that customer. To maximise the system service performance, you are required to implement a scheduling algorithm using multi-level queue strategy with two queues: a high priority Queue 1 and a low priority Queue 2. Queue 1 has absolute priority over Queue 2. In other words, customer request in Queue 2 will only be processed if there is no customer in Queue 1. There is a threshold (=3) that is used to determine whether a customer should remain in Queue 1 (priority number $\leq$ threshold) or Queue 2 (priority number $>$ threshold). Detailed actions in these two queues are listed below:

Queue 1: This is the high priority queue(priority number $\leq$ threshold). Customers in this queue are treated in the way of combined *Highest Priority First* (HPF) and *Weighted Round Robin* (WRR — definition see below) on priority as follows: Select the highest priority customer

Figure 1: Stadium Map.

(customers with the same priority are processed in their arrival order), and then process this customer's request for a ticket quota of $N = \frac{weighted\_time\_quantum}{5 \text{ time units}}$ tickets non-preemptively, then move this customer to the end of (the priority-subqueue of) Queue 1. Here we assume:

- $weighted\_time\_quantum$ = (8 − customer's priority number) × 10 time units.
- 1 ticket costs 5 time units to process.
- customer's priority number is the customer's current priority number.

Weighted Round Robin (WRR): Given n processes $P_1, P_2, \ldots, P_n$, where process P_i has a weight w_i $(1 \leq i \leq n)$, WRR allocates P_i a weighted time quantum of w_i time units (i.e. a share of $\frac{w_i}{\sum_{i=1}^{n} w_i}$ CPU time). In this assignment, to simplify the implementation, a customer's weight is (8 − customer's priority number), and customer's priority number is the customer's current priority number.

Customers of the same priority are processed in their arrival order. The priority of a customer in this queue is decreased by one every 3 runs of this customer, i.e. when a customer has been processed 3 times under its current priority, its priority number will be increased by 1. When a customer's priority number goes above the threshold (=3), it will be demoted from Queue 1 to Queue 2.

For example, the priority number 2 in this queue is decreased by one after 3 runs of this customer, i.e. once booked the first $N = (8 - 2) * 10/5 = 12$ tickets its priority will become 3 and its ticket quota for its 4th run will book $N = (8 - 3) * 10/5 = 10$ tickets (instead of 12 in its first 3 runs).

Queue 2: This is the low priority queue (priority number > threshold). Customers in this queue are handled in *Round Robin*. That is, select the customer with *First Come First Serve*, regardless of their priorities, and process this customer's request in round robin for a fixed time quantum of 20 tickets (= 100 time units) preemptively, then move this customer to the end of Queue 2. The preemption is made by any customer arriving in or promoted to Queue 1, in addition to the time-up preemption in round robin.

Note: Once a running customer X in this queue is interrupted by a new customer (arriving in or promoted to Queue 1), customer X will leave his/her time quantum immediately and the new customer will get the CPU to run. If the priority number of a customer in this queue goes equal to or below the threshold (=3) by the following Ageing mechanism, that customer is promoted from Queue 2 to Queue 1.

Ageing mechanism: Because Queue 1 has priority over Queue 2, and the HPF strategy is applied, starvation may occur. Namely, some customers in Queue 2 may never get to run because there is always a job with higher priority in Queue 1. To address the starvation issue, you must implement a mechanism which ages each customer after 8 runs of other customers. That is, if a customer has waited 8 runs (the interrupted customer is counted as one run) of other customers since his/her last run, his/her priority number will be decreased by one. In this way, the priority of each customer in Queue 2 increases gradually in proportion to the waiting time since the customer's last run.

Note:

- **Order of new arrivals, preemption and promotion**
  - For Queue 1, there may be three types of customers with the same priority i arriving simultaneously at the end of (the priority-i subqueue of) Queue 1:
    - a new arrival customer A,
    - a customer B having completed its weighted time quantum (preempted by Weighted Round Robin),
    - a promoted customer C from Queue 2 (to the priority-3 subqueue of Queue 1).

    Their execution orders in Queue 1 are:
    - For priority-i subqueue, $1 \leq i \leq 2$: priority-i subqueue $\rightarrow A \rightarrow B$;
    - For priority-3 subqueue: priority-3 subqueue $\rightarrow A \rightarrow B \rightarrow C$.
  - For Queue 2, there may be two types of customers arriving simultaneously at the end of Queue 2:
    - a new arrival customer A,
    - a customer B having completed its time quantum (preempted by Round-Robin).

    Their (execution) order in Queue 2 is: $Queue\,2 \rightarrow B \rightarrow A$.
    Note that a demoted customer from Queue 1 is regraded as a customer B, because $Queue\,1 \rightarrow Queue\,2$ demotion and Queue 2 RR preemption cannot occur simultaneously.
- **Order of simultaneous arrivals of same-priority customers:**
  For customers arriving at the same time with the same priority to both queues, we assume their arrival order follows the increasing order of their customer IDs.

Test

Input

We have M customers in the input file. Each customer is identified by a line that describes the customer ID, arrival time (divisible by 5), priority, age and the total tickets required. For example a1 5 1 0 50 describes a customer a1 which arrives at time 5 with priority number 1 and age 0, and requires 50 tickets. One ticket processing consumes 5 time units.

Output

The output includes M+1 lines providing the information of each customer's execution. First line is a string "name arrival end ready running waiting". The next M lines (sorted by termination time) report each customer's ID, arrival and termination times, ready time (the first time the system processes his/her 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/2019/s2/os/assignment1

- Then, check out this directory and add your files:
  svn co https://version-control.adelaide.edu.au/svn/axxxxxxx/2019/s2/os/assignment1
  cd assignment1
  svn add TicketBooker.cpp
  svn add StudentFile1.cpp
  svn add StudentFile2.cpp
  ...
  svn commit -m "assignment1 solution"

- Next, go to the web submission system at:
  `https://cs.adelaide.edu.au/services/websubmission/`
  Navigate to 2019, 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:
  g++ TicketBooker.cpp -o TicketBooker
  ./TicketBooker input.txt>output.txt