CS 1520 / CoE 1520: Programming Languages for Web Applications (Spring 2012)

Department of Computer Science, University of Pittsburgh

(Optional) Assignment #4: Javascript

Released: April 14th, 2012 **Due:** 11:59pm, Tuesday, April 24th, 2012

Goal

Gain familiarity with Javascript.

Description

You are asked to implement in Javascript an interactive visualization of how task scheduling works, under the FIFO (First-In First-Out) scheduling policy. You will be given a list of tasks with specific order, arrival times, and execution cost. Your program will keep an internal "clock", that starts at zero and simulates time in the system. At every time-tick, you need to determine which task should be executing and which one should wait, and make the appropriate updates in the status of all tasks in the system. Time advances when the user clicks a button. When all tasks have finished executing, an `alert()` message should notify the user accordingly.

Input

The list of tasks will be given as input to a form textfield. It will have the following format: *task_number arrival_time execution_cost*, as in the following example:

```
1 2 9
2 5 1
3 8 3
4 15 5
5 20 2
6 20 1
```

The input will be well-formatted and the following assumptions will hold (i.e., you do not need to test them):

- each line contains information about a single task
- all numbers are integers
- *task_number* (i.e., the first column) starts at 1 and is increasing by one after every line
- *arrival_time* (i.e., the second column) of one task is equal or greater to that of the task before it
- *execution_time* (i.e., the third column) is equal or greater than 1

The execution will start after the user clicks the submit button next to the form textfield.

Task Display Box

You should visualize every task in the system as a small window that has the following elements (Fig 1):

- Task number

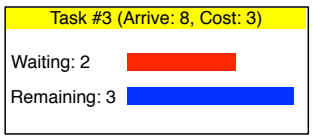

Figure 1: Rough sketch of a task display box

- Wait time – this is how long the task has been waiting in the system (past its arrival time). You should display the actual number and also have a horizontal bar visualizing this.
- Processing time remaining – this is how long the task (still) needs to execute. You should display the actual number and also have a horizontal bar visualizing this. Make the bar a different color than that for the wait time.

Queues

There are four vertical queues in the system (which contain individual task display boxes), shown side by side, as follows (Fig 2):

1. **Input Queue:** This is where all the tasks start. All tasks in this queue have a wait time of 0. This should be the leftmost queue.
2. **Arrived Queue:** This is where all the tasks are moved to when they are considered to have arrived in the system (i.e., system time = arrival time). Tasks in this queue have their wait time updated with every time tick. This should be to the right of the Input Queue.
3. **Executing Queue:** This is where a task is moved when it gets to execute (i.e., the one that is next in order in the arrived queue after the previously executing task has finished execution). The wait time of this task is not changing during execution, but its processing time remaining counter gets decreased (by 1) with every time tick. There is at most one task in this queue at any point of time. This should be to the right of the Arrived Queue.
4. **Output Queue:** This is where a task is moved after it finishes executing (i.e., its processing time remaining counter has been zeroed). Its task display box is "frozen", since none of the counters is updated while in the output queue. This should be to the right of the Executing Queue.

In all queues, tasks should be ordered by *task_number*, with the smallest task number being on the top of the queue.

Figure 2: Rough sketch of the four queues (Note: Task information is not accurate)

Interaction

After the user clicks the start execution button next to the input form textfield, the four queues are displayed and all the tasks are placed in the Input Queue. In addition to the queues, there is a large-font "clock" on the top of the page, that starts at 0, and next to it there are three buttons: +1, +2, and +5, which advance time accordingly.

You should implement the advancement of time (i.e., updating the clock, updating the place of task display boxes, and updating the information in the task display boxes) as a javascript function (and call it the appropriate number of times, according to the button pressed).

Every task needs to have one task display box, with its information. Tasks start in the Input Queue and move their way from left to right, until they reach the Output Queue. When all tasks are in the Output Queue, your program should display an `alert()` message, informing the user.

What to submit

Name your program `scheduling.html`. You can break off your program in multiple files (e.g., .js), to increase readability. Make sure that all paths are **relative**. Submit all files as a single zip file.

Academic Honesty

The work in this assignment is to be done *independently*, by you and only you. Discussions with other students on the assignment should be limited to understanding the statement of the problem. **Cheating in any way, including giving your work to someone else, will result in an F for the course and a report to the appropriate University authority for further disciplinary action.**

How to submit your assignment

We will use a Web-based assignment submission interface. To submit your assignment:

- Go to the class web page `http://db.cs.pitt.edu/courses/cs1520/spring2012` and click the Submit button.
- Use your pittID as the username and the password you specified at the contact information form for authentication. There is a reminder service via email if you forgot your password. You must have already submitted your contact information, if you have not yet you need to do so now.
- Upload your assignment file to the appropriate assignment (from the drop-down list).
- Check (through the web interface) to verify what is the file size that has been uploaded and make sure it has been submitted in full. **It is your responsibility to make sure the assignment was properly submitted.**

You must submit your assignment before the due date (11:59pm, Tuesday, April 24th, 2012) to avoid getting any late penalty. The timestamp of the electronic submission will determine if you have met the deadline. There will be no late submissions allowed after 11:59pm, Thursday, April 26th, 2012.

[Last updated on April 14, 2012 at 2:34am EST]