Printer Simulation: FIFO

Prerequisites, Goals, and Outcomes

Prerequisites: Students should have mastered the following prerequisite skills.

- *Inheritance* - Declaring and defining derived classes
- *Queues* - Use of STL `queue` adapter

Goals: This assignment is designed to reinforce the student's understanding of queues and their use of the STL `queue` adapter.
Outcomes: Students successfully completing this assignment would master the following outcomes.

- Use of the STL `queue` adapter in creating a simulation
- Use inheritance appropriately to create a specialized version of an existing class

Background

From store-and-forward queues in network routers to the facilitation of breadth-first searches in graph algorithms, queues have many important applications in Computer Science and Information Technology. One such application can be found in a policy used by networked printers to manage print jobs. A complicated policy involves the prioritization of jobs that come from certain users. A simpler approach is a first-in-first-out policy. This policy dictates that print jobs are processed in the order they are received. A queue is used to implement this first-in-first-out policy.

Description

This assignment tests your understanding of queues and your ability to use the STL `queue` adapter. The program you are asked to finish the implementation of simulates a shared printer. This printer uses a first-in-first-out queue.

The simulation works by reading and processing a list of events from a data file. Each line in a valid data file contains information about a print job and a time when this job was submitted. Specifically, the information contained in each line is the time (in seconds) the job was submitted, the length in pages of the job, and the name of the computer from which the job was submitted. At the beginning of the simulation, each of these events should be read in by the program and stored in the inherited `workload` queue.

The program should simulate the passage of time by incrementing a counter inside a for-loop or while-loop. The program should initialize this counter to zero and increment it by one second. A print job "arrives" when the current time of the simulation equals the submitted time of the print job at the front of the `workload` queue. When this happens, pop this event from the `workload` queue and place it in another `queue<event>` object. This other `queue<event>` object stores the "arrived" print jobs. These jobs wait in this queue while the program simulates the printing of other jobs. Hence, you may want to name this object `waiting` or something similar.

Files

Following is a list of files needed to complete this assessment.

- *fifo.zip* contains all of the following necessary files:
  - *main.cpp* - Includes function `main`
  - *simulator.h* - Declaration of class `simulator`
  - *simulator.cpp* - Definition of class `simulator`
  - *event.h* - Declaration of class `event`
  - *event.cpp* - Definition of class `event`
  - *job.h* - Declaration of class `job`
  - *job.cpp* - Definition of class `job`
  - *arbitrary.run* - Data file containing arbitrary print jobs
  - *arbitrary.out* - Output from a sample solution when run using *arbitrary.run*
  - *bigfirst.run* - Data file containing larger jobs first
  - *bigfirst.out* - Output from a sample solution when run using *bigfirst.run*

Tasks

To complete this assessment, you need to declare and implement class `fifo`.

To begin, verify the files needed for this assessment.

1. **Extract** the archive to retrieve the files needed to complete this assessment.

Following is an ordered list of steps that serves as a guide to completing this assessment. Work and test incrementally. Save often

1. **First**, declare your class `fifo` in a file named *fifo.h*. Declare class `fifo` appropriately to model the following relationship: a `fifo` is a type of `simulator`.
2. **Next**, complete the implementation of `fifo::simulate`. This member function should first load the data file using the inherited `loadworkload`. Then it should implement the simulation as described above. Use the inherited `seconds_per_page` data member to help determine how long a print job takes to print. Your solution's output should match the output from the sample solutions.

   For clarity, *latency* is the number of seconds that elapse between when a print job arrives and when it begins printing. *Aggregate latency* is the total latency of all print jobs, and *mean latency* is the average latency across all print jobs.

Submission

Submit **only** the following.

1. *fifo.h* - your class `fifo` declaration
2. *fifo.cpp* - your class `fifo` definition