Computer Science 205

Project #4

The Job Queue

Due Date : Friday, December 8th, 11:59 PM

50 Points

Objective

The purpose of this program is to familiarize yourself with the Queue ADT, pointers, and linked lists.

Assignment Summary

First off, we want to implement the Queue ADT using a circular linked list with a single `lastNode` pointer. You need to complete five method bodies that are not yet implemented : `isEmpty`, `dequeueAll`, `enqueue`, `dequeue`, and `front`. Refer to your class notes and handouts where we implemented the Stack ADT using a linked list for a nice reference. Once completed, you will have a queue class that will be used by a driver program.

Now, in many computer systems, jobs are submitted for execution in a batch format. The operating system maintains a queue of job control blocks, each of which is a record storing certain information about a particular job.

Our job control blocks will contain the following information :

- **job identifier** (a string object to represent the job name)
- **job arrival time** (integer)
- **job start time** (integer)
- **expected execution time** (integer)
- **job wait time** – the difference between the start time and arrival time (integer)
- **job turnaround time** – the difference between the arrival time and the finish time (integer)

Write a program to simulate the operation of this system. Your program should read a sequence of job control blocks (ordered according to time of arrival), storing them in an input queue until it is time for them to be processed.

A simulated clock should be used. It should be initialized at 1, and then incremented by 1 throughout your program at appropriate times.

Once the arrival time for a job is reached, place it into a process queue where it remains until completion. Once a job is completed, update its job control block, and place it in an output queue which will be used to print a summary of all jobs.

Keep track the time units when the CPU is idle (no jobs are being processed) and when the CPU is in use. Print the average wait time, total cpu time, total idle time, and cpu usage
(usage time / (usage time + idle time)) to the screen.

General Control Logic for Processing Jobs

After building your input queue, initializing your clock to 1, and your idle and usage counters to 0, perform the following steps during one clock unit :

1. If the input queue is not empty, check the first record in the input queue. Store it in the job queue and remove it from the input queue if the arrival time is equal to the clock time. Even though the CPU may be currently busy with another job, all jobs should be placed in the job Queue at their stated arrival time. They will all be executed first come first served and only one can run at a time.

2. If your job queue is not empty, check the first record in the job queue to see if any jobs need to be removed at the current clock time. If it has already started and its run time is equal to the clock time minus start time, it's time to remove it from the job queue. If you remove a job from the job queue, update its turnaround time, remove it from the job queue, and place it on the finish queue.

3. If your job queue is not empty, check the first record in the job queue. If its arrival time is less than or equal to the clock time and it hasn't already been started, start your job. Update its start time to the clock time, and set its wait time.

4. Update your idle counter, if your job queue is currently empty and your input queue is not. Otherwise, if your job queue is not empty, update your usage counter.

5. Increment your clock counter by 1.

6. Continue this process while the input queue is not empty or the process queue is not empty

Input File and Start Times

Your input file will contain three tokens separated by space on each line. These tokens represent the job name, the arrival time, and the run time. The job name will not contain any spaces. For example,

```
job1 01 03
job2 02 02
job3 10 05
```

You'll be reading each token into a field of a job record, and then inserting that job record into an input queue. Be sure and set your start time to a flag value of something like -1 to indicate that it has not yet been started. This will come in handy during your execution so you'll easily know whether a job has been started or not. You should continue these activities until EOF is reached in your input file.

Sample Output

Your program should print :

- a description of each job processed : the job name, the job arrival time, the expected run time, the actual start time, the job wait time before beginning execution, and the job turnaround time
- the average wait time, the cpu usage, the cpu idle time, and the cpu usage (using a %)

On the next page is a sample run using the input shown above.

```
Job Control Analysis : Summary Report

job id     arrival   start     run    wait   turnaround
           time      time     time    time   time
------     -------   -----    ----    ----   ----------
job1           1        1      3        0        3
job2           2        4      2        2        4
job3          10       10      5        0        5

 Average Wait Time => 0.67
         CPU Usage => 10.00
          CPU Idle => 4.00
     CPU Usage (%) => 71.43%
```

Files and Compilation

You may copy over several files for use with this program by typing :

`cp /pub/digh/CSC205/Prog4/* .`

Your driver program should be named `TimeShare`.

There is no keyboard input for this program. All input should come from a file whose name is sent in from the command line. All output should be sent to the screen, or to a separate output file if you so desire. You should make a copy of the sample input file given to you, and create other test files.

Analysis & Design

Complete an object oriented analysis write-up as before starting this program. Include an actual input example (different from mine) with at least five jobs, and the output that should be produced. Show exactly how your average wait time, cpu usage, cpu idle time, and cpu usage (%) were calculated using your input data. Store in `README`.

Hand In

Using typescript, show a sample compilation and run of your client program using the test file shown in this assignment.

Extra Credit : You will receive five bonus points if you hand in this assignment by 11:59 PM on Wednesday, December 6th.