Data Structures Project 1

Kalen McCrea

Scheduler program outline:

- Global ints:
  - Id - This starts out as 0, which is the id of the first job entered, and increments as new jobs are added to the schedule
  - freePoolMax - This is a const int, the number of processors that are available, which will not change as the program runs
  - freePool - This is the current number of processors available, which will change as jobs occupy processors. It is initialized to be equal to freePoolMax
- Job class
  - Each job has 4 fields:
    - Int job_id – The unique id each job is assigned
    - String job_description – Job description that the user inputs
    - Int n_procs – Number of processors used by the job
    - Int n_ticks – The number of ticks that a job has until completion
- Scheduler class
  - Structures:
    - tick_comparator – This is given to the priority queue as the standard by which it will decide which job should be at the top. It takes 2 jobs, compares the number of ticks, and returns the one with the least.
    - proc_comparator – This is an alternative comparator (mostly used for experimental purposes when creating the scheduler) if you want to sort by least number of processors first.
    - priority_queue – This is the wait queue. Jobs are entered into the wait queue, which is sorted by least number of ticks, and popped into the run queue
    - list – This is the run queue. Jobs that are popped from the wait queue are sent here, where they will be run and removed when they are finished.
  - Functions:
    - newJob() – This function takes a job, checks if the number of processors required are less than the maximum number of processors available to the scheduler, and adds the job to the wait queue if this is true.
    - moveToRunQueue() – This function first checks if the number of processors required for the next job is less than the number currently available in the free pool, and if so, it moves the job to the run queue.
    - runQueueEmpty() – This is simply a Boolean that returns true if the run queue is empty. It is used in the main function since it doesn't have access to the run queue itself.
    - tick() – This is the function that basically runs the program. It first asks the user if they would like to add a new job. If so, it prompts the user for the description, number of processors, and number of ticks, and then calls newJob() to add it to

the wait queue. It then 'ticks'. An iterator runs through the run queue, and decreases the n_tick of every job. If after the tick, a job has 0 ticks remaining, the job is removed from the run queue. If the wait queue isn't empty, it then calls moveToRunQueue to move the next job to the run queue.

- main() – This simply creates a scheduler, ticks once to begin the process, and then keeps ticking until the run queue is empty.

Sample run:

```
Would you like to enter a new job? (y/n)
y
Job description? (no spaces)
job0
Number of processors?
3
Number of ticks?
5

Adding job 0 to wait queue

Would you like to enter a new job? (y/n)
y
Job description? (no spaces)
job1
Number of processors?
5
Number of ticks?
3

Adding job 1 to wait queue

Would you like to enter a new job? (y/n)
y
Job description? (no spaces)
job2
Number of processors?
12
Number of ticks?
3

There are not enough available resources for this job

Would you like to enter a new job? (y/n)
n

TICK
```

```
Current run queue:
----------
----------

Moving job 0 to run queue

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
----------
Job 1:
  Description: job1
  Processors: 5
  Remaining ticks: 2
----------

Moving job 0 to run queue

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
----------
Job 1:
  Description: job1
  Processors: 5
  Remaining ticks: 1
Job 0:
  Description: job0
  Processors: 3
  Remaining ticks: 4
----------

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
----------
Job 1 finished
Job 0:
  Description: job0
```

```
  Processors: 3
  Remaining ticks: 4
---------------------

Would you like to enter a new job? (y/n)
y
Job description? (no spaces)
job3
Number of processors?
4
Number of ticks?
3

Adding job 3 to wait queue

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
---------------------
Job 0:
  Description: job0
  Processors: 3
  Remaining ticks: 3
---------------------

Moving job 3 to run queue

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
---------------------
Job 0:
  Description: job0
  Processors: 3
  Remaining ticks: 2
Job 3:
  Description: job3
  Processors: 4
  Remaining ticks: 2
---------------------

Would you like to enter a new job? (y/n)
```

```
n

TICK

Current run queue:
---------------------
Job 0:
  Description: job0
  Processors: 3
  Remaining ticks: 1
Job 3:
  Description: job3
  Processors: 4
  Remaining ticks: 1
---------------------

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
---------------------
Job 0 finished
Job 3:
  Description: job3
  Processors: 4
  Remaining ticks: 1
---------------------

Would you like to enter a new job? (y/n)
n

TICK

Current run queue:
---------------------
Job 3 finished
---------------------

Process returned 0 (0x0)   execution time : 119.283 s
```

As far as outside sources go, the only sources I consulted were about things like syntax, mostly about the priority queue and comparators because I had never implemented this. I believe all of those solutions came from stackoverflow.