CS/IT 200

Lab 8: Priority Queues

Due Date: One week after assigned lab date

Submission

- Submit all code in a single zip file to the Lab 8 assignment folder on Kodiak.

Goal: Simulate job scheduling on a computer with a variable number of CPUs.

Problem

One of the main applications of priority queues is in operating systems—for scheduling jobs on a CPU. In this project you are to build a program that schedules simulated CPU jobs on a computer with a number of CPUs specified at the start of the simulation.

Your program should run in a loop, each iteration of which corresponds to a time slice for the CPU. Each job is assigned a priority, an integer between -20 (highest priority) and +19 (lowest priority), inclusive. From among the jobs waiting to be processed, the scheduler must assign highest priority jobs first to a CPU that is available.

In this simulation, each job will also come with a length value, which is an integer between 1 and 100, inclusive, representing the number of time slices required to process the job. For simplicity, assume that it is not possible to interrupt a job—once it is assigned to a CPU, the job must be completed before another job can be assigned to the same CPU.

Each time slice, the program should output which jobs are running on which CPUs. For example, in a simulation with 4 CPUs where a job is running on 3 of the 4 CPUs at time slice 22, the program should output:

```
22: A, B, -, C
```

where A, B, and C are job names, and the – indicates an empty CPU. Assume the first time slice is #1.

At the end of the simulation, print the number of jobs that were completed.

Write your simulator in `cpu_simulator.py`. The simulator has two formats in which it can run. The program should ask the user which format it should use:

- File format: The user is asked to input a filename. Jobs will be read in from a CSV file, one row of the file per time slice. If no job is added that time-slice, the row will read "No job". Otherwise, the format of the row is: `job_name, length, priority`. The simulation ends when there are no more rows to read from the file.
- Random format: The user will be asked to give additional parameters. First, ask the user if they want to use a seed for the random number generator (see below about seeds). After that, ask the user to enter how long the simulation should go and how probable it is that a new job is created each time slice. Job length and priority are

selected randomly when a new job is created. For simplicity, let names be unique integer IDs. The simulation ends after the user-specified number of time slices.

Random seeds:

Random number generators can be given a seed which dictates how the numbers are generated. In short, if you give the same seed, then the random numbers generated will be exactly the same. This can be very helpful for debugging! In Python, you can set a seed using the `random.seed()` function as shown below. The value given as an argument must be an integer.

```
seednumber = input('Enter the seed: ')
random.seed(int(seednumber))
```

If the user specifies that they do not want to use a seed, then you must not call this function.

Sample input/output:

```
Enter simulation format, (F)ile or (R)andom: F
Enter number of CPUs: 3
Enter filename: cpu_sim.csv
1: A, -, -
2: A, -, -
3: A, B, -
...
Total number of jobs completed: 17

Enter simulation format, (F)ile or (R)andom: R
Enter number of CPUs: 4
Do you want to use a random seed? Y
Enter the seed: 6095
Enter probability that new job is created each time-slice (0-1): 0.2
Enter length of simulation: 100
1: 1, -, -, -
2: 1, 2, -, -
3: 1, 2, -, -
4: -, 2, -, -
5: 3, 2, -, -
...
Total number of jobs completed: 4
```

Actual sample input/output are provided with this lab. You will be graded on different test data.

What to Submit:

- Your code (`cpu_simulator.py` and any other necessary modules, including modules given as class materials)