Lab Exercise 2: Process State Diagram Simulator

COMP3499 Operating Systems for Engineers
Electrical and Computer Engineering
Wentworth Institute of Technology

Objectives:

- To create a simulator for the Process Model with Two-Suspend States.
  - Understand the Process Model with Two-Suspend States.
  - Understand what each state means.
  - Relate events to state transitions.
  - Create a visual representation of the states of several processes in a system.

Consider a problem similar to the example done in class of the ProcessQuestion.pdf or Chapter 3 slide 45. Design a simulator in C[1] for the process model with two suspend states. For example, given the `inp1.txt` file uploaded on Brightspace (partially shown below):

```
P1 Ready P3 Running P5 Ready P7 Ready P8 Ready end
At time 5:  P3 requests the disk; P7 is dispatched.
At time 15: Time slice for P7 expires; P8 is dispatched.
At time 18: P8 requests the keyboard; P5 is dispatched.
…
```

where the first line lists the initial process states and each line after that shows (an) event(s) that occurs that causes a change in state. You should produce output showing the states of the processes (partially shown below):

```
P1 Ready P3 Running P5 Ready P7 Ready P8 Ready

At time 5:  P3 requests the disk; P7 is dispatched.
P1 Ready P3 Blocked* P5 Ready P7 Running* P8 Ready
disk queue: P3
printer queue:
keyboard queue:

At time 15: Time slice for P7 expires; P8 is dispatched.
P1 Ready P3 Blocked P5 Ready P7 Ready* P8 Running*
disk queue: P3
printer queue:
keyboard queue:

At time 18: P8 requests the keyboard; P5 is dispatched.
```

[1] OS code is written in C and I am giving you some sample code in C to start out with, however, if you prefer C++, you may do this exercise in C++ using objects.

```
P1 Ready P3 Blocked P5 Running* P7 Ready P8 Blocked*
disk queue: P3
printer queue:
keyboard queue: P8

…
```

Therefore, for each line in the input file, update the process states where applicable, print all states marking the updated states with an *, and print the contents of the I/O queues. Assume there are only three I/O queues: disk, printer, and keyboard. Assume that your system has at most 20 states P1 to P20. Processes are added to the appropriate queue when it requests an I/O device and are removed from the queue when the interrupt occurs for that process.

Represent each process with the appropriate composite data type, such as a struct, that must hold the process id (P1 to P20 in this exercise) and the state. Add in any other struct elements that you think are necessary. Represent the collection of processes with the appropriate data structure such as an array of structs, or an array of pointers to structs. Your simulator must be able to accommodate up to 20 structs, but only print the processes being used. Choose the appropriate data structure for each of your queues – an array to hold the process ids is easiest, but the choice is yours.

Your will run your simulator and take screenshots of the output for both input files uploaded to Brightspace. I will run your code on two additional input files (not included on Brightspace) as well for testing purposes. Submit your code as well (`.c` file). See the `LabExercise2_Note.pdf` uploaded on Brightspace.

What to submit:

- A PDF on Brightspace with the names of the team members and screenshots requested above. The screenshots should each have appropriate figure numbers and captions.
- .c files requested above. Do not zip the files together.