COSC 3360/6310 — FUNDAMENTALS OF OPERATING SYSTEMS
COSC 6360 — OPERATING SYSTEMS
ASSIGNMENT #1: ROUND ROBIN SCHEDULING

Now due on Wednesday, October 6, 2021 at 11:59:59 PM

OBJECTIVE

This assignment will introduce you to round robin process scheduling.

SPECIFICATIONS

You are to simulate the execution of a stream of interactive processes by a time-shared system with NCORES processing cores and a very large memory containing enough non-volatile RAM (NVRAM) to replace a conventional disk drive or solid-state drive.

We will assume that we can describe each process by its start time and a sequence of resource requests whose durations are known a priori.

Input Format: Your program should read its input from stdin (C++ cin) and use input redirection as in:

```
$ assignment1 < input1.txt
```

This input will look like:

```
NCORES 2   // system has two cores
SLICE  100 // time slice is 100ms
PROCESS 0  // announcing a new process
START   10 // process starts at t = 10 ms
CORE   200 // 200 ms core request
INPUT   300 // 300ms input request
CORE   200  // 200 ms core request
OUTPUT 10 // write to display for 10 ms
CORE    10 // 10 ms core request
PROCESS 1  // announcing a new process
START 100 // process starts at t = 100 ms
CORE   30  // 30ms core request
...
EOF   // end of input
```

Each process will execute each of its computing steps one by one and ***in the specified order***. Process start times will always be ***monotonically increasing.***

Core scheduling: Your program should schedule core usage according to a ***round-robin policy***: Each process requesting more than SLICE ms of core time will get ***preempted*** and go to the end of the ready queue once it has run for SLICE ms. So, if SLICE equals 100ms and a process requests 140 ms of core time, the process will first get 100ms of core time, then return to the end of the ready queue, and get later the remaining 40ms of core time.

Memory allocation: We assume that memory is large enough to contain all processes plus all system and user files.

I/O requests: We will assume that each process will run in its own window so there will never be any queueing delay. All INPUT and OUTPUT operations will be ***blocking.***

Output specifications: Each time a process arrives or terminates, your program should output a summary report with:

1. The current simulated time
2. The current number of busy cores
3. The contents of the CPU queue
4. For each process in main memory and the process that has just terminated, one line with: the process ID and whether it is in the READY, RUNNING, BLOCKED or TERMINATED state.

Error recovery: Your program can assume that its input will always be correct.

PROGRAMMING NOTES

Your program should start with a block of comments containing your name, the course number, and so on. It should contain functions and these functions should have arguments.

Since you are to focus on the scheduling actions taken by the system you are simulating, your program will only have to act whenever

1. A process is loaded into memory,
2. A process releases a core,
3. A process completes an I/O operation.

You should simulate the flow of time by having a global variable keeping the current time and incrementing it every time you move from one scheduling action to the next.

These specifications were updated on ***Friday, October 1, 2021.*** Check the course respective Teams and Prulu pages for corrections and updates.