COSC 3360/6310— Fundamentals of Operating Systems
Assignment #1 for Spring 2017: Process Scheduling
Now due on Wednesday, February 22 at 11:59:59 PM

Objective
This assignment will introduce you to process scheduling.

Specifications
You are to simulate the execution of a stream of interactive processes by a time-shared system with a very large memory, NCORES processor and one disk drive. We will assume that each process can be described 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
NEW    10 // new process starts at t = 10 ms
CORE    200 // 200 ms core request
DISK    7    // disk access takes 7 ms
CORE    30   // 30 ms core request
DISK    0    // 0 zero-duration disk access
CORE    30   // run for 30 ms
DISPLAY 10 //  write to display for 10 ms
CORE     10 // 10 ms core request
NEW 100 // new process starts at t = 100 ms
CORE  30    // run for 30 ms
...
```

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

Core Scheduling: Your program should schedule core usage according to a ***round-robin policy*** with a time slice of SLICE ms: Each process requesting more than SLICE ms of CPU time will get ***preempted*** and return 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 CPU time, then return to the end of the ready queue, and get later the remaining 40ms of CORE time.

Disk Requests: Zero-duration disk requests correspond to either read operations accessing data that were already in the system I/O buffer or write operations that only involve that buffer. As a result, a process performing a zero-duration disk access will immediately return to the READY state. Other disk requests should be performed in mutual exclusion according to a FCFS (First Come First Served) policy.

Memory Allocation: We assume that memory is large enough to contain all processes.

Display and user input requests: We will assume that each process will run in its own window so there will never be any delay. Both DISPLAY and INPUT operations will be ***blocking.***

Output Specifications: Each time a process terminates, your program should output a summary report with:

1. The total simulated time elapsed
2. The current number of busy cores
3. The current state of the disk unit (busy/idle)
4. The contents of the CPU queue and the disk queue
5. For each process in main memory and the process that has just terminated, one line with: the sequence number of the process, its start time, and its current status (READY, RUNNING, BLOCKED or TERMINATED).

Error recovery: Your program should skip lines that contain unrecognizable keywords.

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 completes a processing step,
3. A process gets preempted after having exhausted its time slice.

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 ***Thursday, February 9, 2017.*** Check the course Piazza pages for corrections and updates.