CISC 310 – Fall 2014
Homework #1: Process Scheduling Simulator

Logistics

- This assignment should be done in C or C++. Your project should compile on a lab machine in CodeBlocks. All necessary files must be included in your submission.
- You can do this assignment in groups of four (or, you know, five or three)
- The assignment is due on **Monday, October 6 at 12:00 NOON via Blackboard** you will need to turn in all your code for the project, as well a PDF document containing your groups answers to the write up questions.
- You will also need to turn in a group work log (please turn in as a PDF file) – this will be discussed in class
- *Your group should also come to class on Monday, October 6 prepared to present your solution to the class. You will want to be able to talk about the design decisions you made, how your program works, and the results that you saw.*

Specifications

You are to simulate the execution of processes by a cheap tablet with a large memory (assume all processes can fit in memory concurrently), one touch screen display, a **dual core** CPU and one solid-state drive (SSD). Each process will be described by its start time followed by a sequence of resource requests. The resources requested will include the CPU (CPU), SSD requests (SSD) and touch screen input/output requests (INP). The input to your program will be a sequence of (request, time) pairs where all times will be expressed in milliseconds. Your program should read input from a file named input.txt located in the same working directory as the program itself.

```
NEW 100       // New process (P0) with start time 100ms
CPU 5         // P0 requests CPU for 5ms
INP 5         // P0 requests I/0 which will take 5ms
CPU 5         // P0 requests CPU for 5ms
NEW 105       // New process (P1) with start time 105ms
CPU 5         // P1 requests CPU for 5ms
```

You can assume there will be no more than 150 pairs and 25 total processes in any file.
You can also assume that memory is large enough to hold all the processes and that context switching times can be ignored.

CPU Scheduling: Your program should have a single ready queue. That queue should be a FIFO queue and should keep all processes ordered according to their queue arrival time in a first come first served order.
SSD: SSD scheduling should be done on a first come first served basis.

If multiple requests are made at the same time you should allocate (1) CPU, (2) SSD, and (3) INP

I/O: Assume that processes never have to wait for the touch screen to become available. However, they cannot be also using a processor or the SSD while handling an I/O event (the process must wait until the I/O is complete)

Your program should have one process table that keeps track of all the processes currently in the system. The process states available are:

- RUNNING = currently executing on a CPU
- READY = waiting for a CPU allocation (in the ready queue)
- WAITING = waiting for INPUT or SSD to finish (or to become available)
- FINISHED = process has finished executing

Every time a process starts, changes state or terminates, your program should print:

a) The current simulated time in milliseconds
b) The states for each process in the process table as well as some information about the current (command) location in each process

When all the processes in your input file have completed your simulator should print a summary report, containing:

a) The total number of processes that have completed
b) The total number of SSD accesses
c) The average duration of an SSD access (including the wait time in the SSD queue)
d) The total time elapsed since the start of the first process
e) The CPU utilization, that is, the average number of busy cores (between zero and two)
f) The SSD utilization, that is, the fraction of time the device was busy (between zero and one)

Write Up

(1) Draw the state transition diagram for the tablet you are modeling. Make sure you include all possible transitions
(2) Draw (a) diagram(s) of the major structures (queues, arrays, etc) that your group is using to keep track of the different devices and processes. Explain briefly the role of each structure, how you decided on its configuration, and how it is used in the program
(3) Explain (using a diagram if necessary) how your group chose to represent a process. What information do you store on each process and why?
(4) What were the major stumbling blocks your group encountered while working on this project? What were the most difficult design decisions to make? Write a 2-3 paragraph reflection on the process of developing this program.

Sample Output (not showing summary data)

(Please note, this is for informational purposes only. Your output may differ, and may be formatted differently)

INPUT FILE

```
NEW 100
CPU 5
INP 5
CPU 5
NEW 105
CPU 5
```

OUTPUT

```
Time = 0
PROCESS 0: Current State = READY, Next Index = 0
PROCESS 1: Current State = READY, Next Index = 4

Time = 100
PROCESS 0: Current State = RUNNING, Next Index = 2
PROCESS 1: Current State = READY, Next Index = 4

Time = 105
PROCESS 0: Current State = WAITING, Next Index = 3
PROCESS 1: Current State = RUNNING, Next Index = 6

Time = 110
PROCESS 0: Current State = RUNNING, Next Index = 4
PROCESS 1: Current State = FINISHED, Next Index = 6

Time = 115
PROCESS 0: Current State = FINISHED, Next Index = 4
PROCESS 1: Current State = FINISHED, Next Index = 6
```