CS436, Exam #2 **Name:**_____

This is a take home exam that is due FRIDAY of finals week, 11:59pm. THIS IS NOT A TEAM PROJECT; do your own work on all parts of this exam and do not share your work with other students.

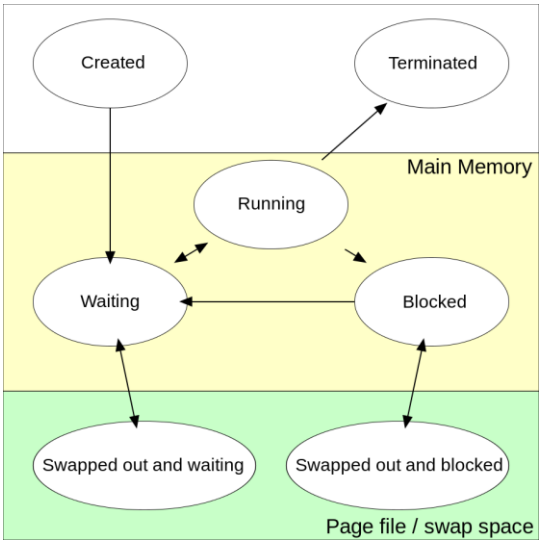
1) Explain/define the abstract concept of a "**System**". Please review this from the readings, and then express a definition in YOUR OWN TERMS/WORDS/IDEAS. Do not just cut and paste this. Further, clarify the differences between **a "system", a "model", and a "simulation"**. (5pts)

2) A key concept of systems thinking is "**Emergence**"; what is this and explain why this is an important concept. Please review this from the readings, and then express your answer in YOUR OWN TERMS/WORDS/IDEAS. Do not just cut and paste this. (5pts)

3) We have discussed system model **verification** and **validation**. Explain these two terms (and how they differ) and give an EXAMPLE of each from your term project. You did not have to actually do these (although even better if you did), **but use your project to explain the difference and how you might have done these**. (10pts)

CS436, Exam #2 **Name:**_____

For the modeling part of Exam #2, you will build a discrete system model of process states in a computer system; recall the process state diagram from your operating systems class. (20pts)



First, a quick review of the OS process state flow from Wikipedia (<u>you can ignore the bottom two states for the purpose of our model</u>):

Created: (Also called New) When a process is first created, it occupies the "created" or "new" state. In this state, the process awaits admission to the "ready" state. Admission will be approved or delayed by a long-term, or admission, scheduler. Typically in most desktop computer systems, this admission will be approved automatically. However, for real-time operating systems this admission may be delayed. In a realtime system, admitting too many processes to the "ready" state may lead to oversaturation and over contention of the system's resources, leading to an inability to meet process deadlines.

Ready: A "ready" or "waiting" process has been loaded into main memory and is awaiting execution on a CPU (to be context switched onto the CPU by the dispatcher, or short-term scheduler). There may be many "ready" processes at any one point of the system's execution—for example, in a one-processor system, only one process can be executing at any one time, and all other "concurrently executing" processes will be waiting for execution. A ready queue or run queue is used in computer scheduling. Modern computers are capable of running many different programs or processes at the same time. However, the CPU is only capable of handling one process at a time. Processes that are ready for the CPU are kept in a queue for "ready" processes. Other processes that are waiting for an event to occur, such as loading information from a hard drive or waiting on an internet connection, are not in the ready queue.

Running: A process moves into the running state when it is chosen for execution. The process's instructions are executed by one of the CPUs (or cores) of the system. There is at most one running process per CPU core.

Blocked: A process transitions to a blocked state when it cannot carry on without an external change in state or event occurring. For example, a process may block on a call to an I/O device such as a printer, if the printer is not available. Processes also commonly block when they require user input, or require access to a critical section which must be executed atomically. Such critical sections are protected using a synchronization object such as a semaphore or mutex.

Terminated: A process may be terminated, either from the "running" state by completing its execution or by explicitly being killed. In either of these cases, the process moves to the "terminated" state. We will only simulate a process completing it's execution and then moving to the terminated state.

Further notes and guidance:

CS436, Exam #2 **Name:**_____

1) In the “Running” state, you can only have as many processes as you have CPU cores. In most common workstations, you would have only 1-8 CPU cores, so create a fixed variable (with a slider) that represents the number of CPU cores and then limits the number of processes in the running state to this number.
2) In the transition from “Running” to “Blocked” which means a running process requested an I/O or other system request, you can simply use another variable with a slider that would be the probability that a process gets blocked during a quantum. Use probabilities from 0.10 to 1.0 and provide a slider for this.
3) Now for the hard stuff. Simulating a quantum of time. If you can figure out how to do this, use 100 milliseconds (1/10 of a second) for your quantum. At the end of each quantum, transition any running process that does not block or terminate back to ready state. Then let InsightMaker randomly select the max number of processes to go back into running state. You could, in the modeling settings select “seconds” as your step and just consider this to represent a quantum since this would be the state change step through the model (a quantum is a common term for the amount of time a process gets the CPU before being evicted in a time-sharing OS). Just state that each step labeled as a second is actually a quantum.
4) More hard stuff. In your initial “pool” of processes (maybe set this to 10-50 range and also use a slider) try to figure out how to assign a random number of quantums that each process instance (an agent) needs before they transition to the terminated state. I would suggest using values ranging from 1 to 100 for the simulation. Then when in the running state, at each step, decrement the individual agents remaining quantum needed by 1, and when it hits 0, it would transition to the terminated state.
5) Please create a line chart that shows the number of “processes” in each state as the simulation runs. Run it long enough for all processes to complete.

To get all of this running in a realistic manner, may be very challenging. So most of the points for this question will be given on the basic model structure (14pts), then (2pts) on if you are able to limit the number of processes in the running state to the value set by a variable with a slider, and then finally the remaining (4pts) on the really hard stuff of simulating quantums and tracking each individual’s quantum needs to move them through the states.

If you can’t figure last part out (quantums), you can just use probabilities for the various transitions This would be a good way to start, then try to figure out the more realistic simulation using needed CPU time (number of quantums). However, using probabilities rather than tracking CPU time needed will give you a max grade of 16/20.

I am giving you a more challenging model to build and simulate (although it does have a very clean definition of transitions and defined states) just to see what different students can come up with. But if you get really stuck, note that I will give the first 14 points (70% grade) for just structuring the states/variables/populations to correctly represent the system. Take your best shot at this and turn in whatever you can come up with.

EXPORT YOUR MODEL (as an InsightMaker file, not a svg file) and submit it along with your PDF file with the answers to the questions in this exam to the Exam #2 submission link in Moodle, Week #11 by FRIDAY finals week, 11:59pm.