Dear Technical Manager

This is a program that simulates a computer system. It contains one “CPU” and two “disks”. This program simulates processes arriving into a computer system, being processed by the CPU and either exiting system, or being processed by one of the disk and being sent back to CPU. The program contains queues or “waiting lines” for each component where the processes wait to use the component. As the program continues to run, new processes continue to arrive and exit program until the time is up and the program is done. To design this program I created three queues, one for each component. There is a CPU queue, a disk1 queue, and a disk2 queue. These queues are first in first out (FIFO) which means the process will be pushed on to the end of the queue and be processed once they get to the front of the queue. I also created one other queue called the event queue. This is a priority queue which means processes are placed in the queue based on their priority. Highest priority processes are at the front of the queue. The priority is based off time, so processes with an earlier time will be ahead of other processes.

To simulate the components of the program such as the CPU, disk 1 and disk 2 I used booleans to portray them. The booleans describe if the component is busy executing a process or if it is empty meaning a process can now be executed on it.

To simulate the state of the process such as when it arrives, when it enters the CPU, ect. I created an enumeration which holds all the state values. This is important because it allows me to access each state through numerical values, which makes the program easier to run.

To simulate each process, I created a structure that contains the state of the process (from the enumeration), the id of the process (what number job it is), and the time that it occurred. Because the priority queue is based on time I created a compare structure which compares each process to be put in the priority queue. If a process has an earlier time it will be placed ahead of the other process. If the processes have the same time the one with the lower job number will be placed ahead of the other. Lastly if they have the same time, and id, they will compare their states. The one with an earlier state will be placed in a higher priority.

Next for each state I create a handler function that deal with state of process and create new processes to be pushed on to queues. Each handle function pops the even off of the priority queue and handles the event by creating new processes and pushing them on to queues. The system runs and continuously uses the handler functions as time advances, until finally the end time is reached and the simulation is over. The handler functions are Job\_arrival, CPU\_enter, CPU\_finish, Disk1\_enter, Disk1\_finish, Disk2\_enter, Disk2\_finish, Job\_exit, and Simulation\_end.

I also created two other functions. One function is a random number generator that randomly generates a random number between the minimum and maximum (which are given in system). This is important because the amount of time a process takes to execute is usually random over a certain interval of time, so this function helps portray the random amount of time at which a process may take to execute, making the program more realistic. The other function I have is a probability select function. There is no way in the simulation to tell when a process will either exit the system or go to the disk, so for this simulation we say that there is a 20 percent chance that the process will exit system. This function randomly chooses a number between 0 and 100 and compares it with the percent given to determine if process will exit system or go to disk, again making the simulation more realistic.

The main method contains the big while loop which the processes move along while the program continues to run. While the priority queue is not empty and the time is not finished the process at the top of the priority queue is executed by one of the handler functions. I have a switch case situation which is grouped by state of the process, so depending on the state of the process determines which handler function it will go to and how it will be handled. In the main function, I also keep track of information that I use for statistical purposes like queue sizes, and component response time. I only use one file to do this program and I keep the component state and queues global so they can be used by all functions throughout the program. I tested my program by printing out everything in main after every iteration of the while loop. I printed out the job number, state, and time, as well as queue sizes (to make sure processes were going to the right disks) and the component state, busy or idle, (to make sure jobs weren’t entering components early or waiting too long before entering a component). Using this information, I debugged and cleaned up my program making sure it worked correctly. I designed this program using C++ to use the queue classes already contained within the language making the program easier to write. I think this program is a great simulation of a computer process, and shows exactly how a computer system runs on the hardware. I hope this is a useful program for the company and is used as a reference for computer systems and how they run.

Sincerely,

Aryanna Anderson