CSCI 4500 Operating Systems
Assignment #3
Due: April 17, 2023

Description

This project will require writing a disk simulation program. Disk accesses are entered into a queue and are then serviced by the disk scheduler. The scheduler will optimize the items in the queue so that they are in a certain order, as we discussed in class. Some of these scheduling algorithms include C-SCAN, FIFO, and SSTF.

The disk you are simulating has 1024 cylinders. The seek time from cylinder to cylinder, and the rotational speed, are as follows:

Distance =	abs(starting cylinder – ending cylinder)
Time =	1 millisecond to start +
	Distance * 0.15 millisecond +
	1 millisecond to stop and settle
Speed =	7200 RPM

We do not care in this assignment about any time needed to switch from head to head in the same cylinder, nor do we care about the data transfer time. But moving from cylinder 100 to, say, 120 would require 1 + 20*0.15 + 1 = 5 milliseconds, plus the rotational latency. **We will use 0.0042 seconds (or 4.2 milliseconds)** for the rotational latency. Going from cylinder 100 to cylinder 120 requires 9.2 milliseconds – the 5 milliseconds to get there and the 4.2 milliseconds for the latency.

Requests for the disk will come in by cylinder number. You will be given a data file (Text format, with one value per line) containing a long list of these numbers, all of which will be between 0..1023. There will be many more data values than the queue size. The cylinder number at the start of the simulation is cylinder zero.

Your program should simulate the C-SCAN, SSTF, and FIFO algorithms as described in the MINIX Book. Suppose that the size of the disk request queue is ten for this example, and that you are simulating C-SCAN. The program would read the first ten requests from the data file and order them according to C-SCAN. Whenever a request is processed, that leaves nine in the queue. At that point read another request to fill it back up to ten, reorder the queue according to C-SCAN, and "seek" again. As the program does the "seek" it should accumulate the total time spent seeking plus add in the rotational latency to get the correct sector.

The goal is to calculate the average time that a request sits in the queue.

Just to make sure we are on the same page, the proper thing to do once the queue is filled is:

- The grand total time spent is set to zero.
- Fill up the queue with requests. Each request has the cylinder number and an accumulated time. Initialize the accumulated time to zero for all the queue entries.
- Repeat:
  - Scan the available offsets in the queue and pick the next appropriate cylinder based on the algorithm.
  - Calculate the time it will take to get to that cylinder, plus the latency once you get there. By the way, a seek from cylinder X to the same cylinder X only adds the latency.
  - Add this time to the accumulated time for all the requests in the queue.
  - Remove the entry you just scheduled and add that entry's accumulated time to the grand total.
  - Read in the next request, initialize its accumulated time to zero, and re-order according to the algorithm.

When the input data file is near the end, the disk queue will have nine, then eight, … down to zero requests, at which point the program is done. The average delay is the grand total divided by the number of requests that you processed.

Process the same data file under the following conditions:

C-SCAN	queue size of 10, 20, 30, 40, 50
SSTF	queue size of 10, 20, 30, 40, 50
FIFO	queue size of 10, 20, 30, 40, 50

The program must accept command line parameters that are:

Algorithm type
Queue size
Name of input file

Bring your data into Excel. Plot a line chart with the queue size on the X axis. The Y axis will have, for each of the three scheduling methods, the average time that it takes to process a disk request. In other words, there will be three plots on this graph.

You may use whatever programming language you prefer for this assignment. However, please comment your code so that I can follow your logic. Upload your code as well as your spreadsheet onto Canvas. I also require some "proof" that the program actually compiles and runs. I will leave that up to you but here are some suggestions:

- A screen shot showing you compiling and running the program.
  Or
- A video showing your code executing.

To recap, you will upload onto Canvas:

- Program & source code
- Spreadsheet with the graphs and data used for the graphs
- Use the file named: **OfficialTestFile.txt** as input to your program to create the Spreadsheet graphs.
- Proof of program execution