CS 3100 Lab 6: Disk Scheduling
Persistence

In Lab 6, you will create a disk scheduling simulator. Your program `disksked` will accept a starting track number and eight tracks to read. `disksked` will simulate FCFS, STFS, SCAN and C_SCAN and produce a report of which tracks will be read in which order and the amount of head movement for each track.

Learning Objectives

- Demonstrate understanding of the four disk scheduling algorithms documented in the text.
- For each scheduling algorithm, correctly output (in order) the tracks to be read, the amount of head movement for each track read and the total head movement for all eight tracks.

Assignment

Begin by installing the tools you will need for this assignment:

```
mkdir -p ~/cs3100/lab6
cd ~/cs3100/lab6
scp USER@icarus.cs.weber.edu:/var/classes/cs3100/lab6/cuke.tar .
tar xvf cuke.tar
```

The file `cuke.tar` file will create, when extracted, a small number of folders containing the cucumber scripts and a `Makefile.` You should not be required to modify the Makefile for this assignment.

Create and edit `disksked.c` using the text editor of your choice. Here are the basic requirements for `disksked.c:`

- `Usage: ./disksked START T1 T2 T3 T4 T5 T6 T7 T8`
    - `START` is the current track of the disk read-write head.
    - `T1-T8` are tracks to be read. Your `disksked` must process exactly eight track values.
    - All track values tested for grading will be integers between 0 and 256
    - All track values are read by your program from the commandline as shown above.
- You are to emulate the following disk scheduling algorithms with the following modifications where noted:
    - First Come First Served (FCFS)
        - Tracks are processed in the order received, from the starting track to each of the 8 tracks in order on the command-line from left to right.
    - Shortest Seek Time First (SSTF)
        - Tracks are processed from the absolute value of the shortest distance from the current track, beginning with the starting track.
        - If there are two tracks that are equal distance from the current track, choose the first one (the earliest one in the list) and move there first.
    - SCAN
        - Re-order the tracks, so that from the current track, move to higher numbered tracks

    first, then reverse to the lower tracks. Process all of the higher numbered tracks in ascending order, then move to the lowest numbered track and process the lower numbered tracks in ascending order.
  - Do not move to track 0 or to the end of the disk during your scan. Some of you might recognize this variant of the SCAN algorithm as the LOOK algorithm.
- Circular SCAN (C-SCAN)
  - Re-order the tracks, so that from the current track, move to higher numbered tracks first in ascending order (as in SCAN), but then reverse direction and process the lower numbered tracks in descending order.
  - Some might recognize this variant as the C-LOOK algorithm.
- Generate a report that contains four lines, in this order: FCFS, SSTF, SCAN and C_SCAN.
  - For each algorithm simulated, output (all on one line):
    - the algorithm name followed by a colon and a space
    - "Start:" followed by the starting track number and a space
    - the eight tracks in order of processing, each followed by a colon and the count of tracks needed to move to that track, each followed by a single space
    - "Total:" and the number of total tracks moved during the simulation.

Sample Output

```
tcowan@tcowan-Ubuntu:~/cs3100/lab6$ ./disksked 53 98 183 37 122 14 124 65 67
FCFS: Start:53 98:45 183:85 37:146 122:85 14:108 124:110 65:59 67:2 Total:640
SSTF: Start:53 65:12 67:2 37:30 14:23 98:84 122:24 124:2 183:59 Total:236
SCAN: Start:53 65:12 67:2 98:31 122:24 124:2 183:59 14:169 37:23 Total:322
C_SCAN: Start:53 65:12 67:2 98:31 122:24 124:2 183:59 37:146 14:23 Total:299
tcowan@tcowan-Ubuntu:~/cs3100/lab6$
```

Files

Here is a list of the files created or modified in this lab:

```
disksked.c
```

Please add your name, lab # (6) and class # (CS 3100) as comments at the top of `disksked.c`.

Additional Requirements

- Please write this program as simply as possible. Do not add any prompts, output or any file I/O except as specified. It is ok for you to generate debugging information while you are testing but use an environment variable as a flag so that I don't see any extraneous output when I run your program. Ask about this if you have questions. Alternatively, simply remember to remove all instrumentation prior to turning in your assignment. Your grade will depend on your adherence to these requirements.
- For this assignment only, I will not grade them until after the semester ends, so you will have no chance to resubmit your assignment because you disregarded a requirement.
- Upload only `disksked.c` to Canvas for grading as you turn in this assignment.

Grading

I use Cucumber scripts to help grade programs. Please note the following:

- The cucumber tests will fail if you do not follow the naming instructions, so please be sure that the programs are named correctly. As long as the source file is named correctly, and you have to upload it to Canvas more than once, the displayed name may have a number after it. That is just for display purposes and your file should still be named correctly internally.
- Upload `disksked.c` to Canvas as you turn in your assignment, when you are ready for me to grade your work.
- It is not necessary to upload any files to icarus for this assignment.
- As always, you are free to deviate from the requirements for this assignment and you agree that if you do, you will receive a zero for the assignment.

NOTE: Just because cucumber reports a certain number of points for your submission of the assignment, the instructor reserves the right to inspect your code, run your submission with additional tools and manually grade your assignment. The decision of the instructor is final. Test your code thoroughly to ensure all requirements are fully met.

Here is how you earn points for this assignment:

FEATURES	POINTS
Must-Have Features	
Only the following file is uploaded to Canvas: `disksked.c`	10
`disksked.c` compiles without errors or warnings	10
`disksked` correctly implements the FCFS scheduling algorithm as documented in this lab.	20
`disksked` correctly implements the SSTF scheduling algorithm as documented in this lab.	20
`disksked` correctly implements the SCAN scheduling algorithm as documented in this lab.	20
`disksked` correctly implements the C-SCAN scheduling algorithm as documented in this lab.	20
Grand Total	100