Programming Project 06
Disk Scheduling Simulator
CS 441/541 – Fall 2017

Project Available		Nov. 28
Component	**Points**	**Due Date (at 11:59 pm)**
Notification of Group Participation		Nov. 30
Project 6		Dec. 12
Program Correctness	85	
Style & Documentation	10	
Testing	5	
Project Total	100	

Late submissions will be subject to the late policy in the syllabus. Your project must be written in C (not C++). No credit will be given for projects written in any programming language other than C. Your project must be able to be compiled using a Makefile by typing the command `make` without any additional arguments in the project directory.

You may work in groups of at most two. Use the D2L groups feature to create your group by the deadline above.

Packaging & handing in your projects

You will turn in a single compressed archive of your completed project directory to the appropriate Autolab project. The compressed archive should be downloaded from the BitBucket website, which is a `.zip` archive. If, for some reason, you cannot download the source from the BitBucket website then turn in a manually compressed archive of the project as a zip file.

Overview

In this project, you will implement a disk scheduling simulator. You will be reading from a file the following information: number of cylinders in the disk, number of cylinder requests, and the specific cylinder being referenced for each of those requests. Command line parameters will pick the file name, set the current position of the head, and current direction of the head.

The following command line argument(s) can occur in **any order** on the command line. All other command line options can be considered file inputs. Your program needs to only handle processing the first file encountered, but be aware that other command line parameters (e.g., `-h`) may occur after the file name.

- Required: `-h #`
  The current head position. The head position can range from 0 to one less than the number of cylinders in the disk defined in the file. If this parameter is not provided then your program will print an error and exit. If this parameter is out-of-range then your program will print an error and exit. Note that you will need to read the file before checking the bounds of this command line option to properly check that the value is in bounds.

- Required: `-d #`
  The current direction of travel of the disk head. If the number is `0` then the disk head is moving toward cylinder `0`. If the number is `1` then the disk head is moving toward the last cylinder on the disk. If the number is neither `0` not `1` then your program will print an error and exit.

You will need to implement the following disk scheduling algorithms.

- **First-come, first-served (FCFS)**
- **Shortest-seek-time-first (SSTF)**
- **SCAN**
- **C-SCAN**
- **LOOK**
- **C-LOOK**

File Format

The file format provides all of the information for the references that you will need to run these algorithms - the file describes the disk queue for I/O to blocks on cylinders.

The first line of the file contains a single positive integer (32-bit) specifying the number of cylinders in the disk. Note that the cylinder numbers in the disk will range from 0 to the number of cylinders minus 1. This is your range of valid cylinder request values.

The second line of the file contains a single positive integer (32-bit) specifying the number of cylinder requests in the disk queue. The number of cylinder requests will range from 1 to an undefined upper bound (your program will need to handle a large disk queue).

The subsequent lines in the file identify the cylinder requests in the disk queue.

The order of the cylinder requests in the file is their ordering in the disk queue. Meaning that the third line of the file (after the two sentinel lines, described above) will be the first request in the disk queue, followed by the second, then third, then .... Do not assume that all cylinders are referenced in the file, and cylinder requests may be repeated in the file. In the example below, the cylinder requests form a queue in the following order: `1000, 86, 0, 1420, 5`.

```
5000
5
1000
86
0
1420
5
```

Example Output

After your program parses the command line arguments, your program will display these parameters. This should identify the input filename, number of cylinders on the disk (specified in the file), the current head position, the current head direction.

After reading the test file, you will **not** need to display the reference string (you may want to do this in testing, but the code you submit should not display the file contents).

Your program will then **run all of the algorithms** in the order previously specified keeping track of the total amount of head movement required by each algorithm.

All of the output will need to be prefixed with the `#` symbol at the front of the line followed by a single space. See the example(s) (below) for information about the ordering of the algorithms and the expected output formatting.

```
shell$ cat given-tests/level1.txt
200
8
98
183
37
122
14
124
65
67
shell$ ./scheduler -h 53 -d 0 given-tests/level1.txt
#------------------------------
# Queue File          : given-tests/level1.txt
# Num. Disk Cylinders : 200
# Head Position       : 53
# Head Direction      : Toward 0
#------------------------------
#   640          FCFS
#   236          SSTF
#   236          SCAN
#   386          C-SCAN
#   208          LOOK
#   326          C-LOOK
shell$ ./scheduler -d 1 -h 53 given-tests/level1.txt
#------------------------------
# Queue File          : given-tests/level1.txt
# Num. Disk Cylinders : 200
# Head Position       : 53
# Head Direction      : Toward last cylinder
#------------------------------
#   640          FCFS
#   236          SSTF
#   331          SCAN
#   382          C-SCAN
#   299          LOOK
#   322          C-LOOK
```

```
shell$ cat given-tests/level2.txt
50
10
21
4
21
4
45
23
16
3
9
42
shell$ ./scheduler given-tests/level2.txt -h 1000 -d 1
Error: Must supply an integer argument greater than or equal to 0 and
       less than the number of cylinders on the disk for the -h option
       This disk has 50 cylinders
shell$ ./scheduler given-tests/level2.txt -h 0 -d 1
#-------------------------------
# Queue File          : given-tests/level2.txt
# Num. Disk Cylinders : 50
# Head Position       :  0
# Head Direction      : Toward last cylinder
#-------------------------------
#   194          FCFS
#    45          SSTF
#    45          SCAN
#    45          C-SCAN
#    45          LOOK
#    45          C-LOOK
shell$ ./scheduler -h 0 given-tests/level2.txt -d 0
#-------------------------------
# Queue File          : given-tests/level2.txt
# Num. Disk Cylinders : 50
# Head Position       :  0
# Head Direction      : Toward 0
#-------------------------------
#   194          FCFS
#    45          SSTF
#    45          SCAN
#    95          C-SCAN
#    45          LOOK
#    87          C-LOOK
shell$ ./scheduler -h 0 given-tests/level2.txt
Error: Must supply the -d option
shell$ ./scheduler given-tests/level2.txt
Error: Must supply the -h and -d options
shell$ ./scheduler -h
Error: Must supply an integer argument greater than 0 for the -h option
shell$ ./scheduler -h 0 -d 1
Error: Did not supply a scheduler file
```

Deliverables:

Your project submission is required be written in a **consistent style** according to the **Style Requirements** handout. You will turn in a single compressed archive of your completed project directory to Autolab. Be sure to carefully review the entire specification in detail for the list of requirements.

- ◯ Program must compile with the Makefile provided without warnings or errors.
- ◯ Command line arguments fully supported, and proper error checking in place (e.g., if the user forgets to supply an integer value to the `-h` parameter).
- ◯ Correct implementation of FCFS
- ◯ Correct implementation of SSTF
- ◯ Correct implementation of SCAN
- ◯ Correct implementation of C-SCAN
- ◯ Correct implementation of LOOK
- ◯ Correct implementation of C-LOOK
- ◯ Complete documentation & code styled according to the **Style Requirements** handout.
- ◯ Set of 5 unique tests