INF 551 – Fall 2017

Homework #1

Due: September 22, Friday
100 points

Task: In this problem, you are asked to implement in **Python** the following disk scheduling algorithms: *SSTF*, *SCAN*, **and *F-SCAN (a new algorithm)***. F-SCAN is a variant of SCAN that addresses the potential starvation problem in the SSTF and SCAN algorithms. Recall that in SSTF and SCAN, there is a single queue containing all the requests, including those that are dynamically arriving. F-SCAN instead maintains two queues Q1 and Q2. Before serving the requests, it freezes the queue Q1 which contains all requests currently outstanding. It can moves the head to serve all requests in Q1 just like SCAN. All new requests will be placed into Q2. When it finishes all requests in Q1, it will move all requests in Q2 to Q1. It then starts the "freeze, serve, and move" cycle. F-SCAN is mentioned in the page 11 of reading chapter. Wikipedia has a bit more details too: https://en.wikipedia.org/wiki/FSCAN.

We assume that the platter has 200 tracks, numbered from 0 to 199, where track #0 is innermost. We also *a*ssume that the disk is currently idle. Thus, in *SCAN* algorithm, the disk head will first move in the direction to the track closest to head, among all requests. Furthermore, when F-SCAN finishes the serving of requests in Q1, we assume that the disk head will move at the direction to serve the request in the new Q1 which is closest to the head.

Input: The program takes as input a text file, e.g., *queue.txt*, formatted as follows.

- The first line is the number of the track where the disk head is currently located.
- The second line is the list of requests, separated by commas, with NO spaces in between.

For example, here is the content of *queue.txt* for the example shown in class.

```
50
120,30,60,135,80,20,95,160,70,5
```

For F-SCAN, you may assume that 10 new requests arrive a time. That is, in the first round, Q1 will contain the first 10 requests, while Q2 contains the second 10; in the second, Q1 will contain the scond 10, Q2 has the third 10 requests; and so on.

Output: For each algorithm, output the ordering of requests, the cost, i.e., the total number of tracks to be travelled, and which request waits for the longest time and what is its wait time (measured by the number of tracks the disk head has moved before serving the request).

Your algorithm should output three lines. The first line shows the schedule; the second line is the total cost; and the third line is the request with the maximum wait time and its time.

For example, *python sstf.py queue.txt* will output:

```
60,70,80,95,120,135,160,30,20,5
265
5,265
```

Execution: Use the command line below to execute your scripts.

python <your_script_name>.py <queue>.txt

Submissions: Name your 3 scripts as below and submit to blackboard by the due time. **DO NOT** make them into folder or zip file.

- <FirstName>_<LastName>_sstf.py
- <FirstName>_<LastName>_scan.py
- <FirstName>_<LastName>_f-scan.py

Note: If there is a tie in deciding which direction to go (e.g., two requests with the same distance from the head), we assume that the head will move at the direction to the innermost track.