Problem Set 4

Due: April 8 at 11:59 PM

In this problem set, you are to implement a number of algorithms that manage the "arm motion" of a disk. We will then compare the performance of the various algorithms. Each of the algorithms will be given a file containing a queue size and a string of requests. A request is a string of integers from 0 to 63. Thus, the file

```
# 4
11
34
12
19
20
32
45
46
12
```

specifies a queue size of 4 and request for cylinders 11, 34 and so on. The queue size indicates how many requests will be considered at one time; in this example, the algorithm must first choose between cylinders 11, 34 12 and 19. If it chooses to service the request for cylinder 12, then it must next choose from 11, 34, 19 and 20.

The algorithms you are to implement are

- Shortest seek first
- First-come first-served

- The elevator algorithm

To compare the various algorithms, we need a notion of *cost*. One approximation is to assume that the cost of a request is linear in the distance; that is if the arm moves from i to j, the cost is $|i - j|$. Another possible cost function is $\sqrt{|i - j|}$.

You have to implement in C the following functions:

```
void shortestSeekFirst( float *ssfLinear, float *ssfRoot, char *fileName);
void firstComeFirstServed( float *fcfsLinear, float *fcfsRoot, char *fileName);
void elevator( float *elevatorLinear, float *elevatorRoot, char *fileName);
```

Each of them tries to open the file with the given file name, If the file cannot be opened, then the function sets each of the costs to -1. Otherwise, it simulates the algorithm and computes the cost using the two cost functions.

Test your functions any way you want. However, you must run the functions on the files data0, data1 ... in the directory */u/cs318/4* on phoenix. Also, present in the directory is a sample "driver" routine driver.c which will be used to test your functions. You can use the files in that directory as a starting point.

Submit all the code, makefile and a README file on phoenix. The README file should contain the results of running your code and your conclusions. Also, *coding style counts*. So, please be neat, use mnemonic variable names and comment tricky code.