Disk Scheduling

This homework uses `disk.py` to familiarize you with how a modern hard drive works. It has a lot of different options, and unlike most of the other simulations, has a graphical animator to show you exactly what happens when the disk is in action.

[Note: there is also an experimental program, `disk-precise.py`, included in the download. This version of the simulator uses the python Decimal package for precise floating point computation, thus giving slightly better answers in some corner cases than `disk.py`. However, it has not been very carefully tested, so use at your own caution.]

Let's do a simple example first. To run the simulator and compute some basic seek, rotation, and transfer times, you first have to give a list of requests to the simulator. This can either be done by specifying the exact requests, or by having the simulator generate some randomly.

We'll start by specifying a list of requests ourselves. Let's do a single request first:

```
prompt> disk.py -a 10
```

At this point you'll see:

```
...
REQUESTS [br '10']

For the requests above, compute the seek, rotate, and transfer times.
Use -c or the graphical mode (-G) to see the answers.
```

To be able to compute the seek, rotation, and transfer times for this request, you'll have to know a little more information about the layout of sectors, the starting position of the disk head, and so forth. To see much of this information, run the simulator in graphical mode (-G):

```
prompt> disk.py -a 10 -G
```

At this point, a window should appear with our simple disk on it.

The disk head is positioned on the outside track, halfway through sector 6. As you can see, sector 10 (our example sector) is on the same track, about a third of the way around. The direction of rotation is counter-clockwise. To run the simulation, press the "s" key while the simulator window is highlighted.

When the simulation completes, you should be able to see that the disk spent 105 time units in rotation and 30 in transfer in order to access sector 10, with no seek time. Press "q" to close the simulator window.

To calculate this (instead of just running the simulation), you would need to know a few details about the disk. First, the rotational speed is by default set to 1 degree per time unit. Thus, to make a complete revolution, it takes 360 time units. Second, transfer begins and ends at the halfway point between sectors. Thus, to read sector 10, the transfer begins halfway between 9 and 10, and ends halfway between 10 and 11. Finally, in the default disk, there are 12 sectors per track, meaning that each sector takes up 30 degrees of the rotational space. Thus, to read a sector, it takes 30 time units (given our default speed of rotation).

With this information in hand, you now should be able to compute the seek, rotation, and transfer times for accessing sector 10. Because the head starts on the same track as 10, there is no seek time. Because the disk rotates at 1 degree / time unit, it takes 105 time units to get to the beginning of sector 10, halfway between 9 and 10 (note that it is exactly 90 degrees to the middle of sector 9, and another 15 to the halfway point). Finally, to transfer the sector takes 30 time units.

Now let's do a slightly more complex example:

```
prompt> disk.py -a 10,11 -G
```

In this case, we're transferring two sectors, 10 and 11. How long will it take? Try guessing before running the simulation!

As you probably guessed, this simulation takes just 30 time units longer, to transfer the next sector 11. Thus, the seek and rotate times remain the same, but the transfer time for the requests is doubled. You can in fact see these sums across the top of the simulator window; they also get printed out to the console as follows:

```
...
Sector:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Sector:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
```

```
TOTALS      Seek:  0  Rotate:105  Transfer: 60  Total: 165
```

Now let's do an example with a seek. Try the following set of requests:

```
prompt> disk.py -a 10,18 -G
```

To compute how long this will take, you need to know how long a seek will take. The distance between each track is by default 40 distance units, and the default rate of seeking is 1 distance unit per unit time. Thus, a seek from the outer track to the middle track takes 40 time units.

You'd also have to know the scheduling policy. The default is FIFO, though, so for now you can just compute the request times assuming the processing order matches the list specified via the "-a" flag.

To compute how long it will take the disk to service these requests, we first compute how long it takes to access sector 10, which we know from above to be 135 time units (105 rotating, 30 transferring). Once this request is complete, the disk begins to seek to the middle track where sector 18 lies, taking 40 time units. Then the disk rotates to sector 18, and transfers it for 30 time units, thus completing the simulation. But how long does this final rotation take?

To compute the rotational delay for 18, first figure out how long the disk would take to rotate from the end of the access to sector 10 to the beginning of the access to sector 18, assuming a zero-cost seek. As you can see from the simulator, sector 10 on the outer track is lined up with sector 22 on the middle track, and there are 7 sectors separating 22 from 18 (23, 12, 13, 14, 15, 16, and 17, as the disk spins counter-clockwise). Rotating through 7 sectors takes 210 time units (30 per sector). However, the first part of this rotation is actually spent seeking to the middle track, for 40 time units. Thus, the actual rotational delay for accessing sector 18 is 210 minus 40, or 170 time units. Run the simulator to see this for yourself; note that you can run without graphics and with the "-c" flag to just see the results without seeing the graphics.

```
prompt> ./disk.py -a 10,18 -c
...
Sector:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Sector:  18  Seek: 40  Rotate:170  Transfer: 30  Total: 240
TOTALS      Seek: 40  Rotate:275  Transfer: 60  Total: 375
```

You should now have a basic idea of how the simulator works. The questions

below will explore some of the different options, to better help you build a model of how a disk really works.