1. Compute the seek, rotation, and transfer times for the following sets of requests: -a 0, -a 6, -a 30, -a 7,30,8, and finally -a 10,11,12,13.

-a	Seek	Rotation	Transfer	Total
0	0	165	30	195
6	0	345	30	375
30	80	265	30	375
7,30,8	160	545	90	795
10,11,12,13	40	425	120	585

2. Do the same requests above but change the seek rate to different values: -S 2, -S 4, -S 8, -S 10, -S 40, -S 0.1. How do the times change?

With large -S value means less seek time, the seek time will shorter, and for 7,10,8 if it is large or equal to -S 4, the total time reduce to 435.

3.Do the same requests above, but change the rotation rate: -R 0.1, -R 0.5, -R 0.01. How do the times change?

With long rotation, the total time also increase. Also, the transfer time also increase.

4. FIFO is not always best, e.g., with the request stream -a 7,30,8, what order should the requests be processed in? Run the shortest seek-time first (SSTF) scheduler (-p SSTF) on this workload; how long should it take (seek, rotation, transfer) for each request to be served?

The correct order is 7,8,30 since 7 and 8 on the same track. With SSTF the seek is 80, rotate is 204, and transfer is 90.

5. Now use the shortest access-time first (SATF) scheduler (-p SATF). Does it make any difference for -a 7,30,8 workload? Find a set of requests where SATF outperforms SSTF; more generally, when is SATF better than SSTF?

NO difference for -a 7,30,8. But when seek time is short.
With -a 6, 31 and -S 40, the SATF is doing better than SSTF

6. Here is a request stream to try: -a 10,11,12,13. What goes poorly when it runs? Try adding track skew to address this problem (-o skew). Given the default seek rate, what should the skew be to maximize performance? What about for different seek rates (e.g., -S 2, -S 4)? In general, could you write a formula to figure out the skew?

The sequential read doing poorly, because when head change the track to read 12 and 13, it total miss it. Thus, it need wait entire rotation to read. The skew should be -o 2 to maximize performance. The general formula as follow:

$$\text{Ceiling}\left(\left(\frac{seek\ time}{dist\ per\ sector}\right) mod\ \#sector\right)$$
$$\text{ceiling}((40/30)\text{mod } 12)\ =\ 2$$

-S 2 should be 1, and -S 4 should be 1 as well based on above formula

7. Specify a disk with different density per zone, e.g., -z 10,20,30, which specifies the angular difference between blocks on the outer, middle, and inner tracks. Run some random requests (e.g., -a -1 -A 5,-1,0, which specifies that random requests should be used via the -a -1 flag and that five requests ranging from 0 to the max be generated), and compute the seek, rotation, and transfer times. Use different random seeds. What is the bandwidth (in sectors per unit time) on the outer, middle, and inner tracks?

The outer sector are 135, 270, 140. Then bandwidth for outer is 3/(135+270+140) = 0.0055
The middle is 370, 260 Then the bandwidth for middle is 2/(170+260) = 0.0046

8. A scheduling window determines how many requests the disk can examine at once. Generate random workloads (e.g., -A 1000,-1,0, with different seeds) and see how long the SATF scheduler takes when the scheduling window is changed from 1 up to the number of requests. How big of a window is needed to maximize performance? Hint: use the -c flag and don't turn on graphics (-G) to run these quickly. When the scheduling window is set to 1, does it matter which policy you are using?

Window need be 1000 to have a maximize performance. When the scheduling window is set to 1, it doesn't matter.

9. Create a series of requests to starve a particular request, assuming an SATF policy. Given that sequence, how does it perform if you use a bounded SATF (BSATF) scheduling approach? In this approach, you specify the scheduling window (e.g., -w 4); the scheduler only moves onto the next window of requests when all requests in the current window have been serviced. Does this solve starvation? How does it perform, as compared to SATF? In general, how should a disk make this trade-off between performance and starvation avoidance?

The sequence is -a 30,7,8,9,10 with -p SATF, can starve sector 30, total time is 375
Same sequence with -p BSAT and window 4, won't starve sector 30, but overall performance become 495. With bound window it solves the starve, and the performance reduce. In general, it depend on the sequence of the sectors.

10. All the scheduling policies we have looked at thus far are greedy; they pick the next best option instead of looking for an optimal schedule. Can you find a set of requests in which greedy is not optimal?

./disk.py -a 5,20 -c we have total time of 435, but with ./disk.py -a 5,20 -c -p SATF we have 345