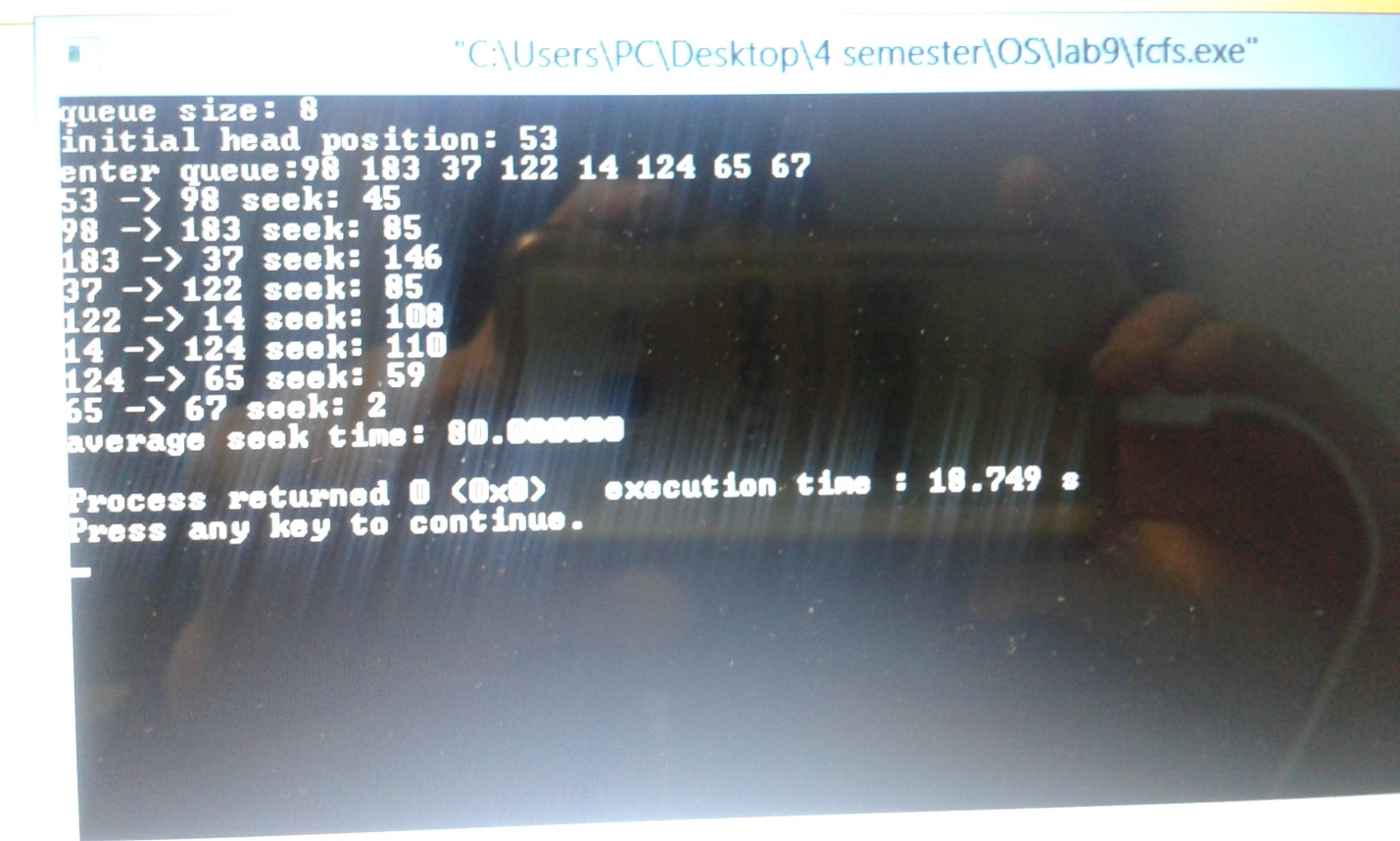
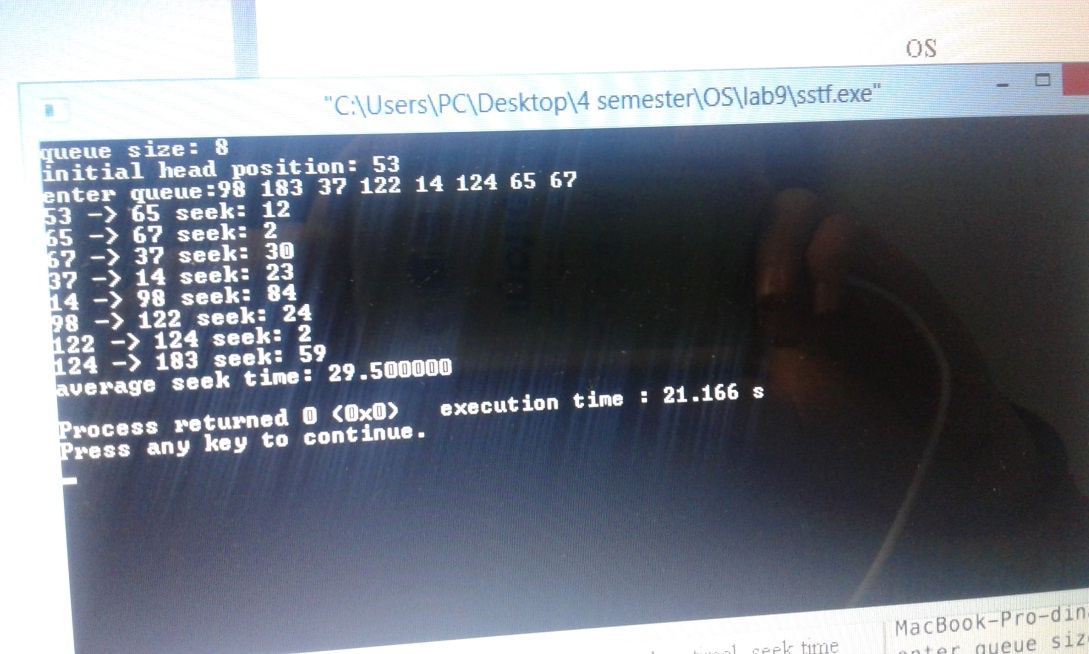
In general disk-scheduling algorithms are used to when the disk queue have several pending requests. So operating system chooses which pending request to service next.

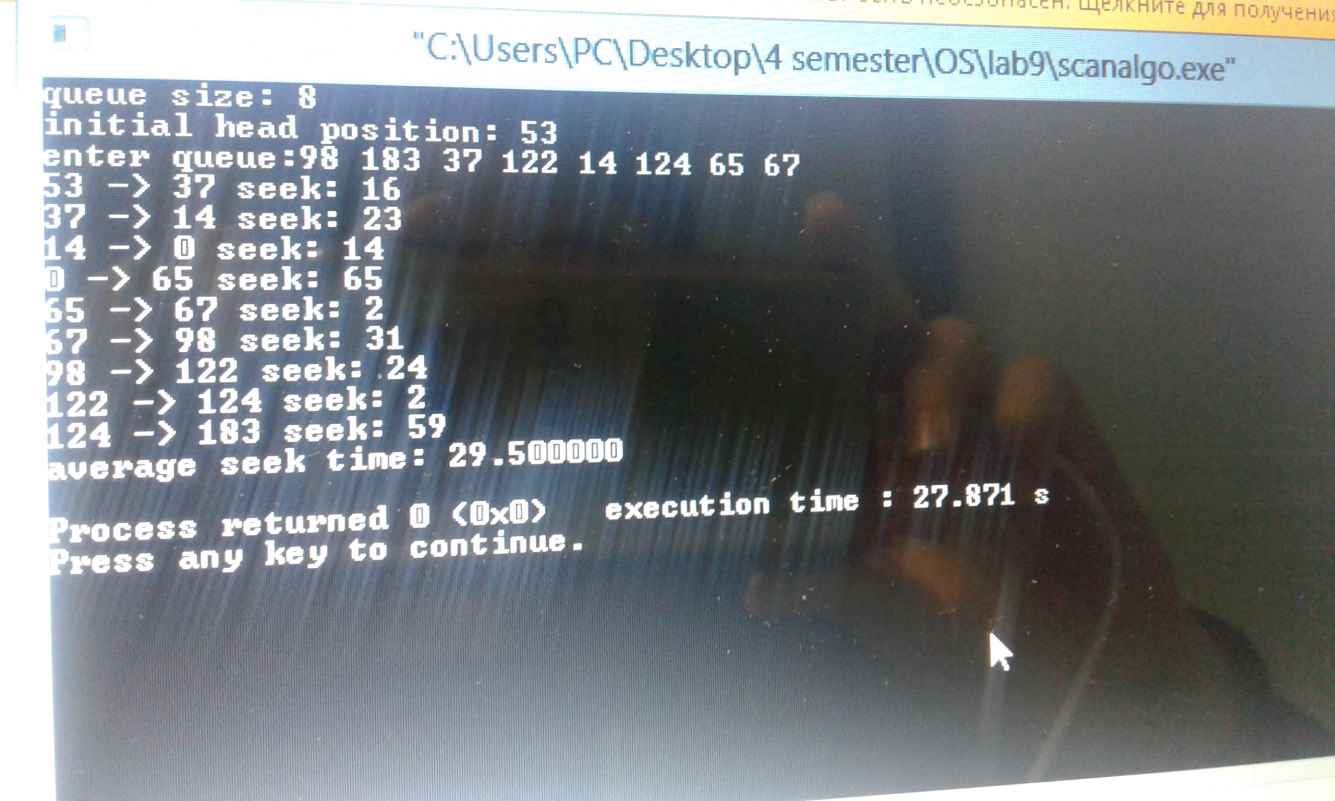
1. FCFS: Here requests are served in the order they arrive in queue, as fifo. Its implementation is the simplest. We have array queue[] containing requests for I/O to blocks on cylinders and integers as head and seek. Seek is time needed to move the heads to the cylinder containing the desired sector. After user entered queue elements we set as its first element our head and following elements will be saved as they came depending under fifo rule. Then final we print path of head movement with each seek time as difference of neighbor elements of queue and print average seek time as a result of division of total seek time to size of queue. However this algo does not try to optimize seek time. Result as shown:



2.SSFS: As its name says here requests which have shortest seek time will be served first. Its implementation is similar to FCFS, but we should to calculate for each element of queue difference with head as we need to choose the pending request closest to the current head position and store them to array d[]. Then we sort elements of this array and our queue elements in ascending order, so finally print result. However this algo may cause starvation for requests with longer seek time compared to incoming ones. Result:



3.SCAN:Also,known as elevator, as here disk arm moves into a particular direction serving all requests in its path then it reverses its direction and again services until the end of queue. In its implementation we have new arrays as big[] and small[] which will store queue elements that are bigger/smaller than our head. Then we compare our queue elements with head and from a result store them to corresponding arrays dynamically incrementing size of these arrays. Then we sort elements of each array: small[]array in descending, big[] array in ascending order. So as our queue will be sorted we set its first element as our head and following we reassign to our userdefined queue array firstly with our new sorted small[] array and it goes to zero,then our big[]array. Then print result. However in this algo we can have long waiting time for requests for locations just visited by disk arm. Result as shown below:



4.CSCAN:Known as circular scan algorithm, as it works almost similar to scan, but instead reversing its direction it goes to other end and starts from there servicing. In its implementation we have to know max element. And when we reassign to our userdefined queue array firstly with our new sorted big[] array till max and it goes to 0 , then our small[]array , which previously will be sorted in ascending order both. Then print result. Result as follows:

