Operating Systems (OS) – Theory

Assignment:01

Scheduling Algorithms (Simulation)

Submitted by:

Tayyaba Asif

2019-CE-5

Submitted to:

Sir Waqas Ali

Date: 11th February, 2022

Question No.01:

Compute the response time and turnaround time when running three jobs of length 200 with the SJF and FIFO schedulers.

Answer:

Here arrival time of every job is considered to be 0. Since job length is same response time and turnaround time will be same in both cases.

Turnaround Time= Exit Time-Arrival Time

Response Time= Time at which the process gets the CPU-Arrival Time

Response Time:

J1=0-0=0

J2=200-0=200

J3=400-0=400

Average response time=$\frac{0+200+400}{3}$=$\frac{600}{3}$=200

Turnaround Time:

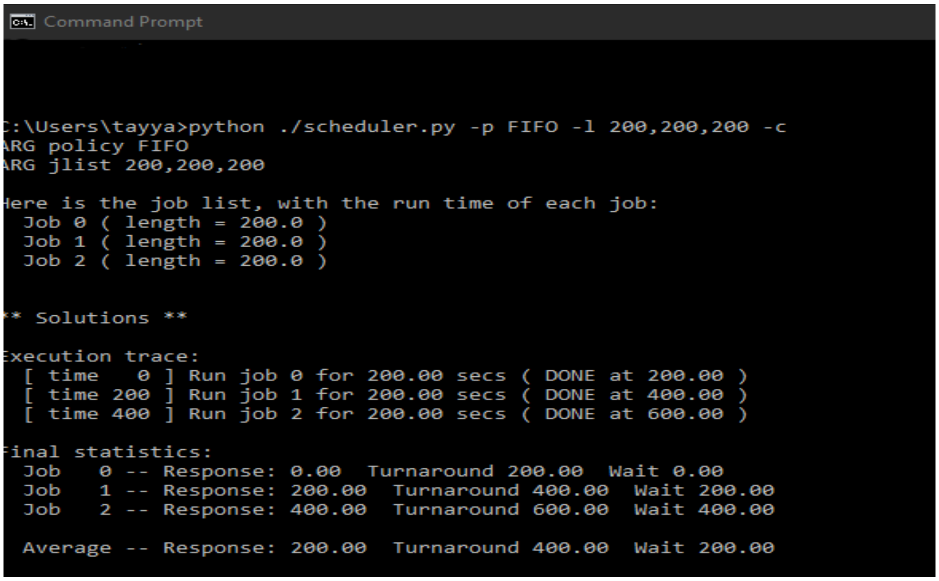
J1=200-0=200

J2=400-0=400

J3=600-0=600

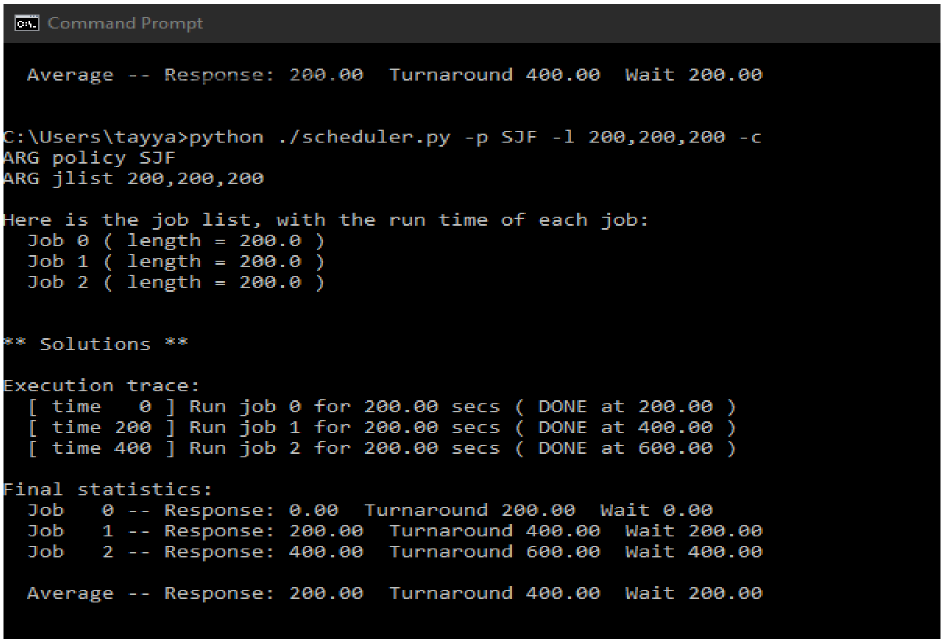
Average turnaround time=$\frac{200+400+600}{3}$=$\frac{1200}{3}$=400

FIFO:



```
C:\Users\tayya>python ./scheduler.py -p FIFO -l 200,200,200 -c
ARG policy FIFO
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 200.00 secs ( DONE at 400.00 )
  [ time 400 ] Run job 2 for 200.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 400.00  Wait 200.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 200.00  Turnaround 400.00  Wait 200.00
```

SJF:



```
  Average -- Response: 200.00  Turnaround 400.00  Wait 200.00


C:\Users\tayya>python ./scheduler.py -p SJF -l 200,200,200 -c
ARG policy SJF
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 200.00 secs ( DONE at 400.00 )
  [ time 400 ] Run job 2 for 200.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 400.00  Wait 200.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 200.00  Turnaround 400.00  Wait 200.00
```

<u>Question No.02:</u>

Now do the same but with jobs of different lengths: 200, 100, and 300.

<u>Answer:</u>

Here arrival time of every job is considered to be 0.

Turnaround Time= Exit Time-Arrival Time

Response Time= Time at which the process gets the CPU-Arrival Time

FIFO (First In First Out):

Response Time:

J1=0-0=0

J2=200-0=200

J3=300-0=300

Average response time=$\frac{0+200+300}{3}=\frac{500}{3}$=166.66

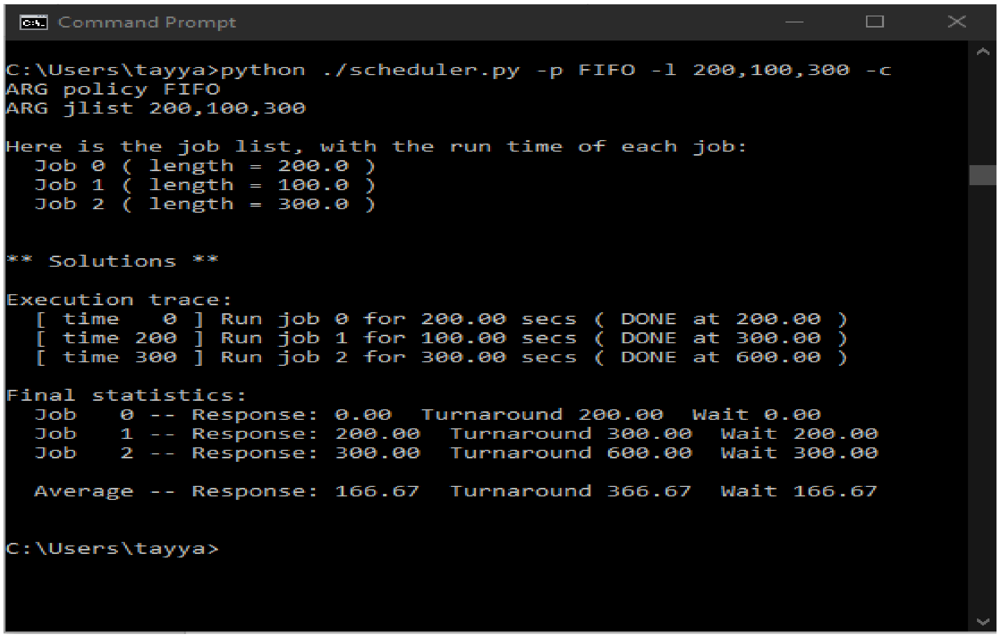
Turnaround Time:

J1=200-0=200

J2=300-0=300

J3=600-0=600

Average turnaround time=$\frac{200+300+600}{3}$=$\frac{1100}{3}$=366.66



```
C:\Users\tayya>python ./scheduler.py -p FIFO -l 200,100,300 -c
ARG policy FIFO
ARG jlist 200,100,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 300.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 100.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 2 for 300.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 300.00  Wait 200.00
  Job   2 -- Response: 300.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 166.67  Turnaround 366.67  Wait 166.67


C:\Users\tayya>
```

SJF (Shortest Job First):

Response Time:

J1=0-0=0

J2=100-0=100

J3=300-0=300

Average response time=$\frac{0+100+300}{3}$=$\frac{400}{3}$=133.3

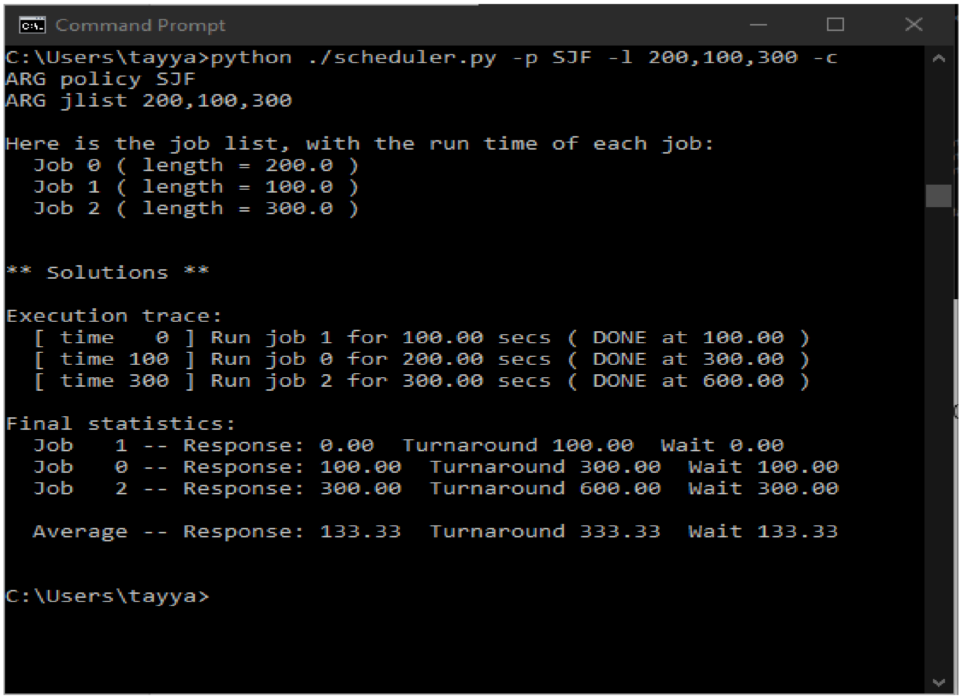
Turnaround Time:

J1=100-0=100

J2=300-0=300

J3=600-0=600

Average turnaround time=$\frac{100+300+600}{3}$=$\frac{1000}{3}$=333.3



```
C:\Users\tayya>python ./scheduler.py -p SJF -l 200,100,300 -c
ARG policy SJF
ARG jlist 200,100,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 300.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 1 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 0 for 200.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 2 for 300.00 secs ( DONE at 600.00 )

Final statistics:
  Job   1 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   0 -- Response: 100.00  Turnaround 300.00  Wait 100.00
  Job   2 -- Response: 300.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 133.33  Turnaround 333.33  Wait 133.33


C:\Users\tayya>
```

Question No.03:

Now do the same, but also with the RR scheduler and a time-slice of 1.

Answer:

Here quantum q is given as 1 so it will take so many iterations to complete the process. We can calculate response time easily but turnaround time cannot be calculated easily with small quantum.

Response Time:

J1=0-0=0

J2=1-0=1

J3=2-0=2

Average response time=$\frac{0+1+2}{3}=\frac{3}{3}$=1

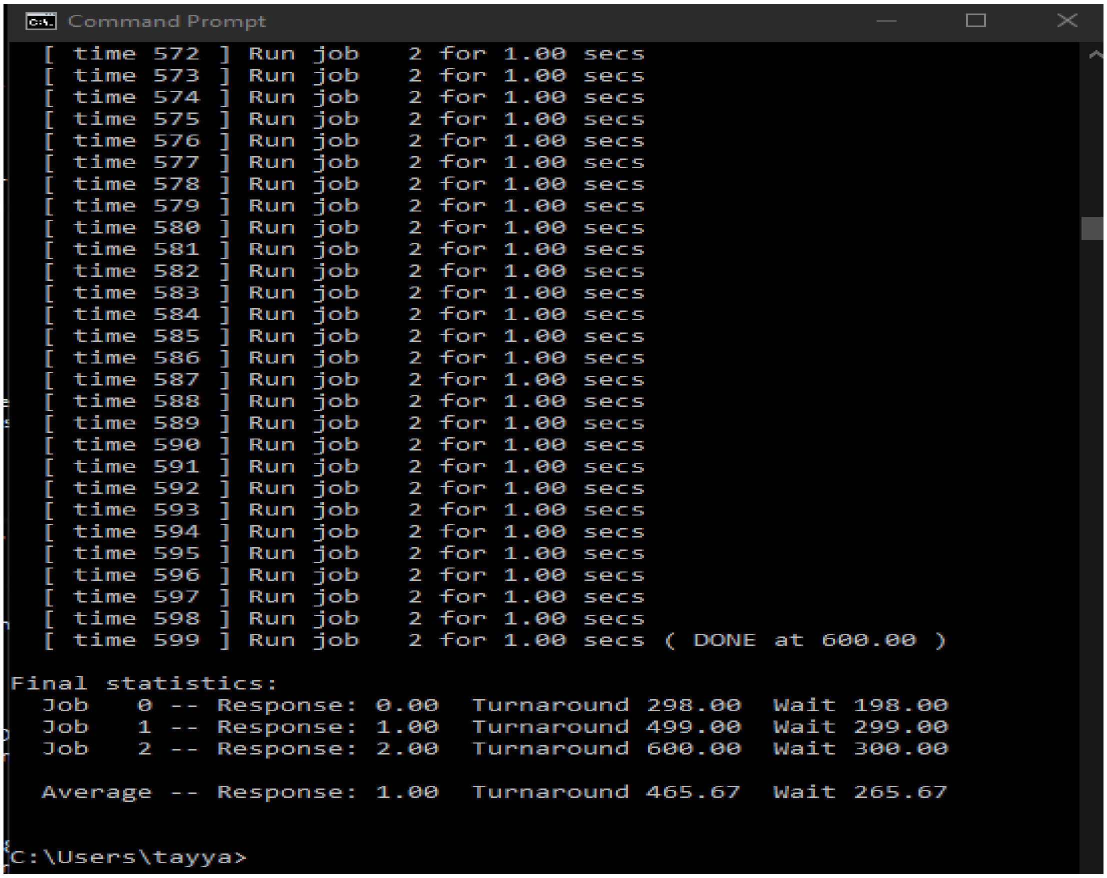
Turnaround Time:

J1=298-0=298 (Exit time is 298 because process gets CPU at 0 and in between switching there are 2 other processes)

J2=499-0=499 (Exit time is 499 because process gets CPU at 0 and in between switching there are 2 other processes in start and after completion of J1 only J2 and J3 are left)

J3=600-0=600 (Exit time is 600 because process gets CPU at 0 and in between switching there are 2 other processes in start and after completion of J1 and J2 only J3 is left)

Average turnaround time=$\frac{298+499+600}{3}$=$\frac{1397}{3}$=465.66



```
  [ time 572 ] Run job   2 for 1.00 secs
  [ time 573 ] Run job   2 for 1.00 secs
  [ time 574 ] Run job   2 for 1.00 secs
  [ time 575 ] Run job   2 for 1.00 secs
  [ time 576 ] Run job   2 for 1.00 secs
  [ time 577 ] Run job   2 for 1.00 secs
  [ time 578 ] Run job   2 for 1.00 secs
  [ time 579 ] Run job   2 for 1.00 secs
  [ time 580 ] Run job   2 for 1.00 secs
  [ time 581 ] Run job   2 for 1.00 secs
  [ time 582 ] Run job   2 for 1.00 secs
  [ time 583 ] Run job   2 for 1.00 secs
  [ time 584 ] Run job   2 for 1.00 secs
  [ time 585 ] Run job   2 for 1.00 secs
  [ time 586 ] Run job   2 for 1.00 secs
  [ time 587 ] Run job   2 for 1.00 secs
  [ time 588 ] Run job   2 for 1.00 secs
  [ time 589 ] Run job   2 for 1.00 secs
  [ time 590 ] Run job   2 for 1.00 secs
  [ time 591 ] Run job   2 for 1.00 secs
  [ time 592 ] Run job   2 for 1.00 secs
  [ time 593 ] Run job   2 for 1.00 secs
  [ time 594 ] Run job   2 for 1.00 secs
  [ time 595 ] Run job   2 for 1.00 secs
  [ time 596 ] Run job   2 for 1.00 secs
  [ time 597 ] Run job   2 for 1.00 secs
  [ time 598 ] Run job   2 for 1.00 secs
  [ time 599 ] Run job   2 for 1.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 298.00  Wait 198.00
  Job   1 -- Response: 1.00  Turnaround 499.00  Wait 299.00
  Job   2 -- Response: 2.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 1.00  Turnaround 465.67  Wait 265.67

C:\Users\tayya>
```

Question No.04:

For what types of workloads does SJF deliver the same turnaround times as FIFO?

Answer:

Whenever the processes arrive at the same time and they have same length both SJF and FIFO returns the same turnaround time as shown in question number 1.

Question No.05:

For what types of workloads and quantum lengths does SJF deliver the same response times as RR?

Answer:

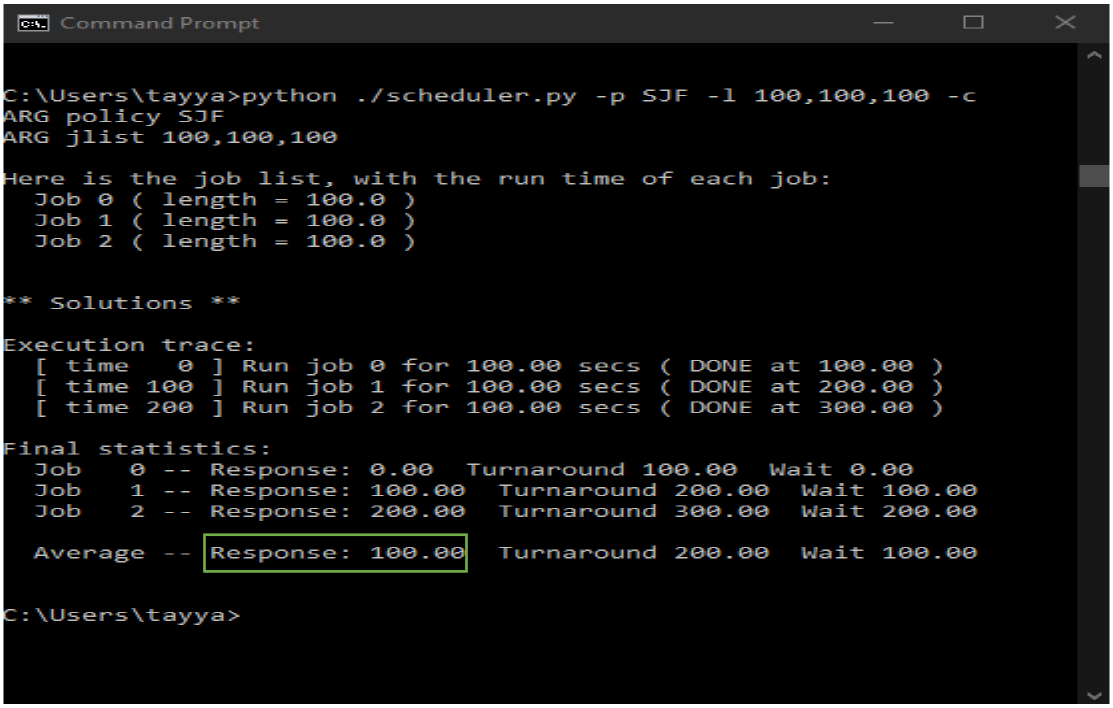
Whenever quantum ‘q’ becomes equal to the length of process SJF and RR returns the same response time.

Question No.06:

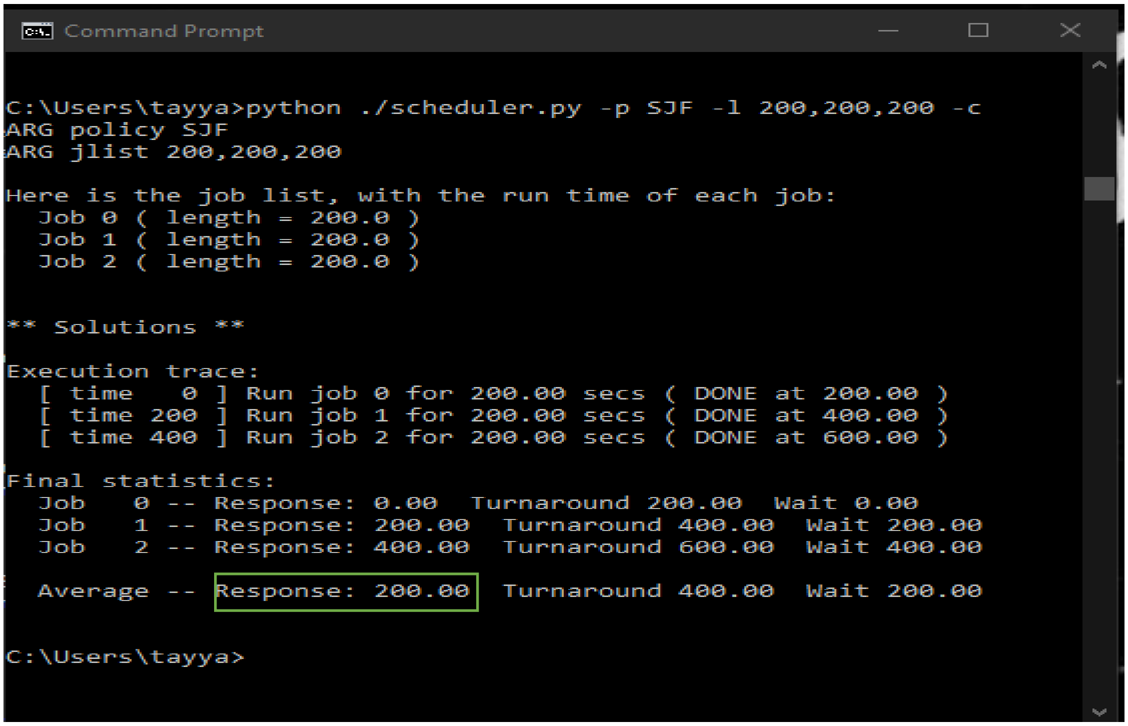
What happens to response time with SJF as job lengths increase? Can you use the simulator to demonstrate the trend?

Answer:

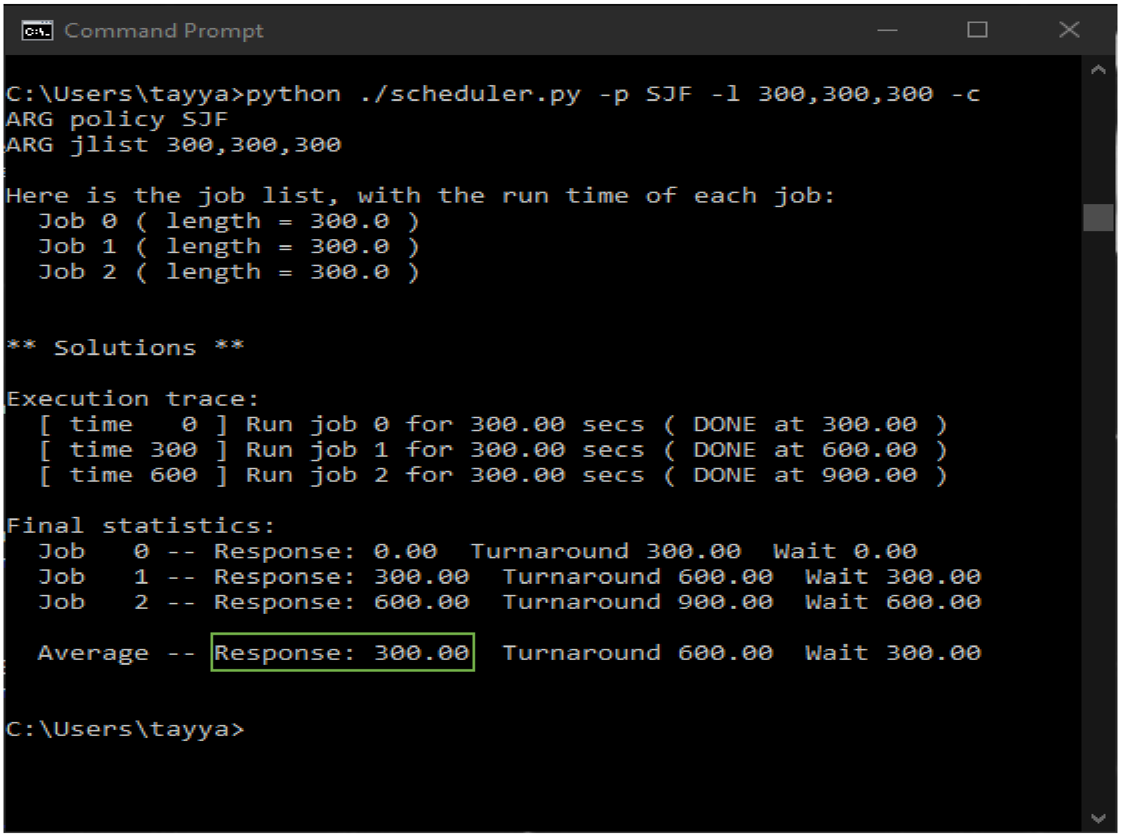
When length of the job increases, the response time also increases. This is shown in the screenshots below.



```
C:\Users\tayya>python ./scheduler.py -p SJF -l 100,100,100 -c
ARG policy SJF
ARG jlist 100,100,100

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 100.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 100.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 2 for 100.00 secs ( DONE at 300.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 200.00  Wait 100.00
  Job   2 -- Response: 200.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 100.00  Turnaround 200.00  Wait 100.00


C:\Users\tayya>
```



```
C:\Users\tayya>python ./scheduler.py -p SJF -l 200,200,200 -c
ARG policy SJF
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 200.00 secs ( DONE at 400.00 )
  [ time 400 ] Run job 2 for 200.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 400.00  Wait 200.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 200.00  Turnaround 400.00  Wait 200.00

C:\Users\tayya>
```



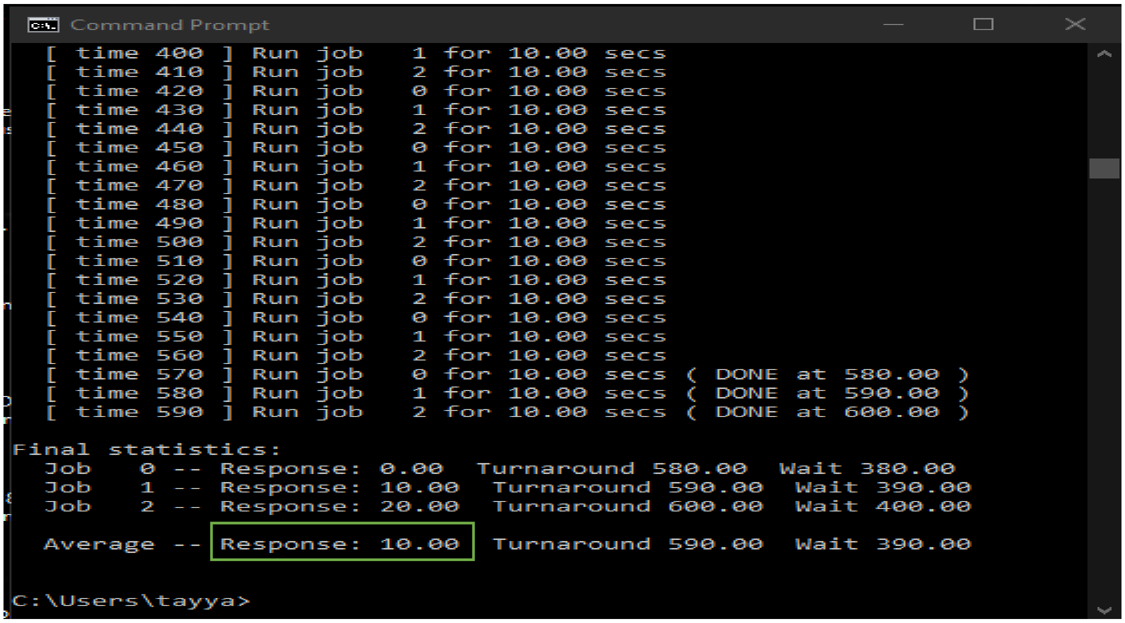
```
C:\Users\tayya>python ./scheduler.py -p SJF -l 300,300,300 -c
ARG policy SJF
ARG jlist 300,300,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 300.0 )
  Job 1 ( length = 300.0 )
  Job 2 ( length = 300.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 300.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 1 for 300.00 secs ( DONE at 600.00 )
  [ time 600 ] Run job 2 for 300.00 secs ( DONE at 900.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 300.00  Wait 0.00
  Job   1 -- Response: 300.00  Turnaround 600.00  Wait 300.00
  Job   2 -- Response: 600.00  Turnaround 900.00  Wait 600.00

  Average -- Response: 300.00  Turnaround 600.00  Wait 300.00

C:\Users\tayya>
```

Question No.07:

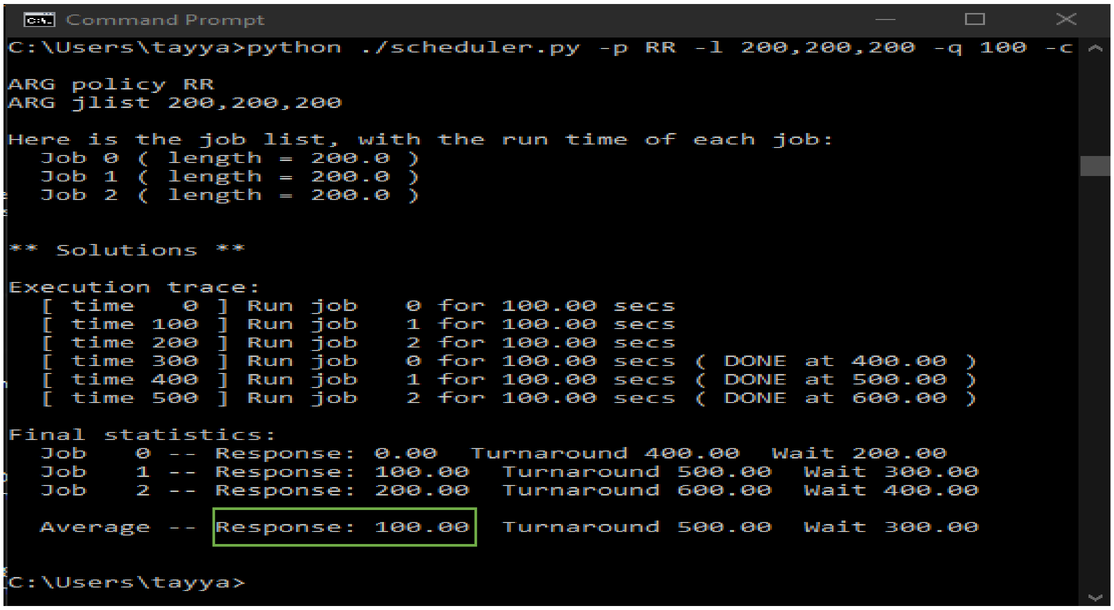
What happens to response time with RR as quantum lengths increase? Can you write an equation that gives the worst-case response time, given N jobs?

Answer:

When quantum length is increased, response time of Round Robin (RR) also increases. For the worst-case response time $\frac{(N-1)*q}{N}$ can be used.



```
  [ time 400 ] Run job   1 for 10.00 secs
  [ time 410 ] Run job   2 for 10.00 secs
  [ time 420 ] Run job   0 for 10.00 secs
  [ time 430 ] Run job   1 for 10.00 secs
  [ time 440 ] Run job   2 for 10.00 secs
  [ time 450 ] Run job   0 for 10.00 secs
  [ time 460 ] Run job   1 for 10.00 secs
  [ time 470 ] Run job   2 for 10.00 secs
  [ time 480 ] Run job   0 for 10.00 secs
  [ time 490 ] Run job   1 for 10.00 secs
  [ time 500 ] Run job   2 for 10.00 secs
  [ time 510 ] Run job   0 for 10.00 secs
  [ time 520 ] Run job   1 for 10.00 secs
  [ time 530 ] Run job   2 for 10.00 secs
  [ time 540 ] Run job   0 for 10.00 secs
  [ time 550 ] Run job   1 for 10.00 secs
  [ time 560 ] Run job   2 for 10.00 secs
  [ time 570 ] Run job   0 for 10.00 secs ( DONE at 580.00 )
  [ time 580 ] Run job   1 for 10.00 secs ( DONE at 590.00 )
  [ time 590 ] Run job   2 for 10.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 580.00  Wait 380.00
  Job   1 -- Response: 10.00  Turnaround 590.00  Wait 390.00
  Job   2 -- Response: 20.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 10.00  Turnaround 590.00  Wait 390.00

C:\Users\tayya>
```



```
C:\Users\tayya>python ./scheduler.py -p RR -l 200,200,200 -q 100 -c

ARG policy RR
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job   0 for 100.00 secs
  [ time 100 ] Run job   1 for 100.00 secs
  [ time 200 ] Run job   2 for 100.00 secs
  [ time 300 ] Run job   0 for 100.00 secs ( DONE at 400.00 )
  [ time 400 ] Run job   1 for 100.00 secs ( DONE at 500.00 )
  [ time 500 ] Run job   2 for 100.00 secs ( DONE at 600.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 400.00  Wait 200.00
  Job   1 -- Response: 100.00  Turnaround 500.00  Wait 300.00
  Job   2 -- Response: 200.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 100.00  Turnaround 500.00  Wait 300.00

C:\Users\tayya>
```