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

```
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
```

python ./scheduler.py -p FIFO -j 3 -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
```

python ./scheduler.py -p SJF -j 3 -l 200,200,200 –c

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

```
ARG policy FIFO
ARG jlist 100,200,300

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


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 200.00 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 100.00  Wait 0.00
  Job   1 -- 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
```

python ./scheduler.py -p FIFO -j 3 -l 100,200,300 -c

```
ARG policy FIFO
ARG jlist 100,300,200

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


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 100.00 secs ( DONE at 100.00 )
  [ time 100 ] Run job 1 for 300.00 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 100.00  Wait 0.00
  Job   1 -- Response: 100.00  Turnaround 400.00  Wait 100.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 166.67  Turnaround 366.67  Wait 166.67
```

python ./scheduler.py -p FIFO -j 3 -l 100,300,200 -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
```

python ./scheduler.py -p FIFO -j 3 -l 200,100,300 -c

```
ARG policy FIFO
ARG jlist 200,300,100

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


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 200.00 secs ( DONE at 200.00 )
  [ time 200 ] Run job 1 for 300.00 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 200.00  Wait 0.00
  Job   1 -- Response: 200.00  Turnaround 500.00  Wait 200.00
  Job   2 -- Response: 500.00  Turnaround 600.00  Wait 500.00

  Average -- Response: 233.33  Turnaround 433.33  Wait 233.33
```

python ./scheduler.py -p FIFO -j 3 -l 200,300,100 -c

```
ARG policy FIFO
ARG jlist 300,100,200

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


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 300.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 1 for 100.00 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 300.00  Wait 0.00
  Job   1 -- Response: 300.00  Turnaround 400.00  Wait 300.00
  Job   2 -- Response: 400.00  Turnaround 600.00  Wait 400.00

  Average -- Response: 233.33  Turnaround 433.33  Wait 233.33
```

python ./scheduler.py -p FIFO -j 3 -l 300,100,200 -c

```
ARG policy FIFO
ARG jlist 300,200,100

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


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 300.00 secs ( DONE at 300.00 )
  [ time 300 ] Run job 1 for 200.00 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 300.00  Wait 0.00
  Job   1 -- Response: 300.00  Turnaround 500.00  Wait 300.00
  Job   2 -- Response: 500.00  Turnaround 600.00  Wait 500.00

  Average -- Response: 266.67  Turnaround 466.67  Wait 266.67
```

python ./scheduler.py -p FIFO -j 3 -l 300,200,100 -c

```
ARG policy SJF
ARG jlist 200,300,100

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


** Solutions **

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

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

  Average -- Response: 133.33  Turnaround 333.33  Wait 133.33
```

python ./scheduler.py -p SJF -j 3 -l 200,300,100 -c

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

```
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
```

python ./scheduler.py -p RR -j 3 -q 1 -l 100,200,300 -c

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 298.00  Wait 198.00
  Job   1 -- Response: 1.00  Turnaround 600.00  Wait 300.00
  Job   2 -- Response: 2.00  Turnaround 500.00  Wait 300.00

  Average -- Response: 1.00  Turnaround 466.00  Wait 266.00
```

python ./scheduler.py -p RR -j 3 -q 1 -l 100,300,200 -c

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 499.00  Wait 299.00
  Job   1 -- Response: 1.00  Turnaround 299.00  Wait 199.00
  Job   2 -- Response: 2.00  Turnaround 600.00  Wait 300.00

  Average -- Response: 1.00  Turnaround 466.00  Wait 266.00
```

python ./scheduler.py -p RR -j 3 -q 1 -l 200,100,300 -c

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 499.00  Wait 299.00
  Job   1 -- Response: 1.00  Turnaround 600.00  Wait 300.00
  Job   2 -- Response: 2.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 1.00  Turnaround 466.33  Wait 266.33
```

python ./scheduler.py -p RR -j 3 -q 1 -l 200,300,100 -c

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 600.00  Wait 300.00
  Job   1 -- Response: 1.00  Turnaround 299.00  Wait 199.00
  Job   2 -- Response: 2.00  Turnaround 500.00  Wait 300.00

  Average -- Response: 1.00  Turnaround 466.33  Wait 266.33
```

python ./scheduler.py -p RR -j 3 -q 1 -l 300,100,200 -c

```
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 600.00  Wait 300.00
  Job   1 -- Response: 1.00  Turnaround 500.00  Wait 300.00
  Job   2 -- Response: 2.00  Turnaround 300.00  Wait 200.00

  Average -- Response: 1.00  Turnaround 466.67  Wait 266.67
```

python ./scheduler.py -p RR -j 3 -q 1 -l 300,200,100 -c

4. **For what types of workloads does SJF deliver the same turnaround times as FIFO?**
   Для тих, у яких тривалість усіх процесів однакова або ж вони прибувають у неспадному порядку тривалості.

5. **For what types of workloads and quantum lengths does SJF deliver the same response times as RR?**
   Для тих, у яких тривалість усіх процесів рівна й дорівнює квантовій довжині або ж вони прибувають у неспадному порядку тривалості, а квантова довжина не менша за найдовший процес.

6. **What happens to response time with SJF as job lengths increase? Can you use the simulator to demonstrate the trend?**
   Він збільшується, адже час початку процесів настає пізніше відповідно до довжини попередніх процесів.

```
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
```

python ./scheduler.py -p SJF -j 3 -l 100,100,100 -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
```

python ./scheduler.py -p SJF -j 3 -l 200,200,200 -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
```

python ./scheduler.py -p SJF -j 3 -l 300,300,300 -c

7. **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?**
   Збільшується, адже зі збільшенням квантової довжини кожен процес довше очікуватиме свого початку, відповідно час відповіді також збільшується.

   Найгірший сценарій тоді, коли квантова довжина не менша, ніж довжина найдовшого процесу. Такий випадок можна виразити наступним рівнянням:

$$T_{response\_worst} = \sum_{i=1}^{N-1} T_{response_i} + t_i,$$

   де $T_{response_i}$ — час відповіді і-ого процесу, t_i — тривалість і-ого процесу.