# Report
# On
# Programming Assignment 2

COEN 346 sect. YJ-X

Date of submission:

March 6th, 2023

By

Tobias Smith 40165892

Magy Gerges 40157151

Alexis Bernier 40208693

NOTE: Due to mismanagement by the team, the FairShare class was not finished hence will not be considered in this lab report. What was done for that class will still be included in the zip file including the source code.

Below are the sample inputs that was given to the four algorithms in the form of a txt file (schedule.txt):

T1, 4, 20

T2, 3, 25

T3, 3, 25

T4, 5, 15

T5, 5, 20

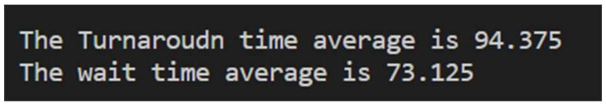
T6, 1, 10

T7, 3, 30

T8, 10, 25

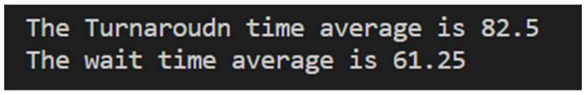
**Results:**

1) FCFS



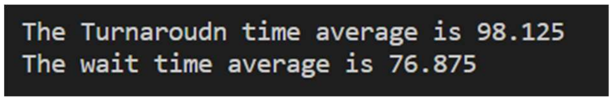
```
The Turnaroudn time average is 94.375
The wait time average is 73.125
```

2) SJF



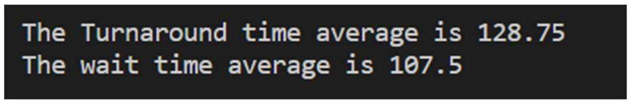
```
The Turnaroudn time average is 82.5
The wait time average is 61.25
```

3) Priority



```
The Turnaroudn time average is 98.125
The wait time average is 76.875
```

4) Round Robin


```
The Turnaround time average is 128.75
The wait time average is 107.5
```

Comparing the above results from the different algorithms, we can see that the average turnaround and average wait time differs greatly from one algorithm to the next. With FCFS, with the given inputs, the wait time seems to be optimized whereas the turnaround time seems to be overshadowed. This algorithm benefits any task that request the CPU the earliest, this means that any task or process that needs to run but are stuck waiting for the ones that arrived before this one. In other words, the issue with this algorithm is that if a later process has higher priority, it will not run until its turn arrives.

With SJF, any job that has a shorter burst time will run before the longer ones, this could potentially lead to starvation where the longer burst processes will never run.

The priority algorithm runs processes that have a higher priority. Similar to SJF, this could also lead to processes with lower priority to spend a long time in the wait queue before they can run.

With Round Robin, all tasks have the same time quantum to run, this means they have in our example 10 time units to run and then go back in the wait queue. The issue with this algorithm lies with the time quantum. With a large time quantum, this would essentially become a FCFS algorithm as all processes would have the time to run their entire burst before it switches to the next queued task. With a small time quantum, this would increase the turnaround time of all processes as they would have little time to run their burst before going back in the queue.

With FairShare, a user holds a certain number of task and have a time quantum that would be split to all its task. This is similar to Round Robin where depending on the time quantum given, this could either harm the turnaround time or the wait time.

For FCFS, with the given inputs, the wait time seems to be optimized whereas the turnaround time seems to be overshadowed. In fact, this is the case for all four algorithms. However, comparing the performance of the above-mentioned algorithms, the clear cut winner for the best average wait time and the best average turnaround time is SJF.

**Contribution Table:**

| | |
|---|---|
| Toby Smith | Wrote most algorithms |
| Magy Gerges | Turnaround and wait time for FCFS, SJF, Priority |
| Alexis Bernier | Turnaround and wait time for FCFS, SJF, Priority, RR, Debugging and testing all algorithms, Report, (had to fixe RR class) |