MLFQ Report

My MLFQ implementation is here: https://github.com/ZackHu-2001/MLFQ

Implementation details

The program is split into four parts, the Process class, the Queue Class, the main.cpp, and a helper class Configuration that is used to read in configuration. Where in main.cpp have a main loop, which handles the overall logic of the MLFQ scheduler.

Unaccomplished task

For right the moment, the MLFQ has achieved the basic rules for MLFQ. However, it does not reach my expectations: handle some other extreme scenarios. Which is due to the high workload of this week. CS5010 has an incredibly high workload this week, and the workload for CS5600 is getting higher this week, thus there are still unaccomplished tasks, but I would fix it within these days.

Testing result

Some cases are tested manually:

```
./MLFQ -n 3 -q 10 -A 10,10,10 -b 50 -L 0,20,0;10,30,0;5,50,0 -s 10 -Q 10,20,50
```

```
./MLFQ -n 3 -q 10 -A 10,10,10 -b 50 -L 0,80,0;10,120,0;5,150,0;150,100,0 -s 10 -Q 10,20,50
```

```
./MLFQ -n 3 -q 10 -A 10,10,10 -b 50 -L 0,80,5;10,120,10;5,150,20 -s 10 -Q 10,20,50 -i 5
```

```
./MLFQ -n 3 -q 10 -A 10,10,10 -b 50 -L 0,80,1;10,120,2;5,150,3 -s 10 -Q 10,20,50 -i 5
```

Their behavior is within the expectation of MLFQ and shows the same result as the MLFQ.py provided in the textbook.

Comparison with other teammates

Until 4:30 Monday, only one of the teammates had published his code. Therefore, in this report, I will only compare my implementation with his implementation.

From language level:

Both my implementation and his are based on OOD, however, his implementation is written in Rust, while mine is in C++.

From the implementation level:

I think right at the moment, we both only finished the basic rules mentioned in MLFQ, without adding further rules to handle extreme scenarios.