ECE 3058
Thread Scheduling Lab

Name: Dinyar Islam
GT Username: dislam6

Problem 1B

When 1 CPU,
# of Context Switches: 99
Total execution time: 67.6 s
Total time spent in READY state: 389.9 s

When 2 CPU,
# of Context Switches: 113
Total execution time: 37.0 s
Total time spent in READY state: 84.5 s

When 4 CPU,
# of Context Switches: 180
Total execution time: 33.9 s
Total time spent in READY state: 0.8 s

From the above data, we can see that is general when the CPU increases, the execution time decreases, but the relati
onship is not
linear. The decrease in CPU execution time decreases by 30.6 s when CPU number increases from 1 to 2, but decrea
sed a lot less only
by 3.1 s when CPU number increased from 2 to 4. This is because when you increase the number of CPUs, its going
to make the execution time better until the threshold point
is reached where the CPUs start to be under utilized as they are not used/required. In other words, when the degree o
f multiprogramming is beyond the maximum amount necessary,
the CPUs are no longer used to their full capacity. In our case, when we increased the number of CPUs from 2 to 4, t
hat is what happened and we saw a very minimal change in
execution time as we were reaching optimal performance and increasing the number of CPUs will eventually no lon
ger make a noticeable difference.

Problem 2B

When 200ms
# of Context Switches: 363
Total execution time: 67.6 s
Total time spent in READY state: 286.8 s

When 400ms
# of Context Switches: 202
Total execution time: 67.6 s
Total time spent in READY state: 301.2 s

When 600ms
# of Context Switches: 160
Total execution time: 67.6 s
Total time spent in READY state: 315.0 s

When 800ms
# of Context Switches: 135
Total execution time: 67.6 s
Total time spent in READY state: 327.7 s

As seen above, the shortest time spent waiting was at the 200ms time slice and as the time slices increased so did the waiting time, beacause
the total time spent in ready queue will decrease with shorter time slices.

In a real OS, the shortest timeslice is not always the best choice because the CPU will be too busy switching between processes to
complete jobs which means a lot of time is spent switching contexts.

Problem 3B

While it is easy to simulate an LRTF algorithm in the simulator, it is essentially impossible to implement precisely in real life
because in a real os we usually don't know exactly how much time remains in a process and the time remaining tends to constantly change.
However, in our simulation we are running it for fixed amounts of time, and in real life thats quite impossible becaus e the algorithm
is not able to predict the amount of time remaining ahead of execution.

When using FIFO:
# of Context Switches: 99
Total execution time: 67.6 s
Total time spent in READY state: 389.9 s

When using Round Robin 200ms:
# of Context Switches: 363
Total execution time: 67.6 s
Total time spent in READY state: 286.8 s

When using LRTF:
# of Context Switches: 106
Total execution time: 68.6 s
Total time spent in READY state: 452.9 s

As seen above, Round-Robin has the lowest wait time, followed by FIFO and LRTF. This is because for LRTF No o ther process can execute
until the longest job or process executes completely leading to very high average waiting times.