Dor Gross, 039344999 & Ben Reuveni 300095296

Operating Systems - Exercise 4

Part 2:

Algorithm	Average Waiting Time	Average Turnaround Time	Average Response Time
FCFS	5.2	9.4	5.2
SJF	4.2	8.4	4.2
PSJF	2.8	7	1.6
RR	5.4	9.6	2.6

Calculation process (processing time in black, waiting time in blue):

FCFS:

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛													
P2		🟦	🟦	🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛	⬛									
P3				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	⬛								
P4									🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛	⬛	⬛			
P5														🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛

Total Waiting Time: 0+7+9+5+5 Average Waiting Time: 5.2
Total Turnaround Time: 8+11+10+10+8 Average Turnaround Time: 9.4
Total Response Time: 0+7+9+5+5 Average Response Time: 5.2

SJF:

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛													
P2		🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛	⬛								
P3				🟦	🟦	🟦	🟦	🟦	⬛												
P4									🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛	⬛	⬛
P5														⬛	⬛	⬛					

Total Waiting Time: 0+8+5+8+0 Average Waiting Time: 4.2
Total Turnaround Time: 8+12+613+3 Average Turnaround Time: 8.4
Total Response Time: 0+8+5+8+0 Average Response Time: 4.2

PSJF:

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	⬛	🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛	⬛	⬛	⬛	⬛								
P2		⬛	⬛	🟦	⬛	⬛															
P3				⬛																	
P4									🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	⬛	⬛	⬛	⬛	⬛
P5														⬛	⬛	⬛					

Total Waiting Time: 5+1+0+8+0   Average Waiting Time: 2.8
Total Turnaround Time: 13+5+1+13+3   Average Turnaround Time: 7
Total Response Time: 0+0+0+8+0   Average Response Time: 1.6

RR (q=2):

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	⬛	⬛	🟦	🟦	⬛	⬛	🟦	🟦	🟦	⬛	⬛	🟦	🟦	⬛	⬛						
P2		🟦	⬛	⬛	🟦	🟦	⬛	⬛													
P3				🟦	🟦	🟦	🟦	🟦	⬛												
P4									🟦	🟦	🟦	⬛	⬛	🟦	🟦	⬛	⬛	🟦	🟦	⬛	
P5														🟦	🟦	🟦	🟦	⬛	⬛	🟦	⬛

(Note: we first enqueue the old task that was just preempted, and then add any new task)
Queue: 1 2 1 2 3 1 4 1 4 5 4 5

Total Waiting Time: 7+3+5+7+5   Average Waiting Time: 5.4
Total Turnaround Time: 15+7+612+8   Average Turnaround Time: 9.6
Total Response Time: 0+1+5+3+4   Average Response Time: 2.6

Part 3:

A. FCFS is a preemptive scheduling scheme

False. In FCFS, every process is scheduled for its whole burst time, and only then the next process gets its CPU time.

B. Round Robin is optimal for minimizing the context switches overhead

False. In round robin there are many context switches, as processes are being dequeued and enqueued all the time while performing many preemptions.

C. Round Robin is optimal for minimizing the context switches overhead

True. In FCFS there are context switches linear with the number of processes (i.e., for N processes there are N-1 context switches).

D. SJF is always better than (or equals to) PSJF with relation to both response time and waiting time

False. Short processes will have to wait for earlier longer jobs to finish before getting CPU time in SJF, while in PSJF they will get CPU time immediately (lower response time and waiting time).

(Note: The example in part 2 shows exactly the opposite - that PSJF is better than SJF)

E. Interactive systems are likely to use Round Robin as their scheduling scheme

True. Interactive systems requires faster response. Round Robin tries to equal the share of time across all active processes (so no process will have to wait for too long).

Part 4:

A. PSJF satisfies starvation freedom

False. We will give a counterexample.

Consider the following scenario - every millisecond arrives a new process with a demand for 1 millisecond of CPU time, and, in time 0 also arrives a process which demands 3 milliseconds.

Since there's a process requiring only 1 millisecond in every moment, the process which requires 3 milliseconds will never get its CPU time, and will be starved forever.

B. FCFS satisfies starvation freedom

True. Proof using contradiction:

Let $\{p_1, p_2, p_3, \dots\}$ denote an infinite series of processes' requests for CPU time measured in milliseconds.

Contrary assume that there exists at least one process' request which is starved forever, and let p_n be the first such process' request, i.e., p_n will never get its CPU time. Consider the set $P = \{p_1, p_2, \dots, p_{n-1}\}$, and let T be $T = \Sigma_{p \in P}(p)$. According to FCFS, processes are executed according to the time of their arrival, therefore all the processes preceding p_n will have their CPU time (and done) after T milliseconds, which implies that p_n will get its CPU time after T milliseconds, which is a **contradiction** to the assumption that p_n is starved forever.

And therefore, FCFS satisfies starvation freedom. **QED**