Operating Systems

Opdracht 3

5.1

I/O-bound programs have longer I/O burst cycles and shorter CPU burst cycles. CPU-bound programs work the other way around. The scheduler must know if a process is I/O or CPU bound because it can change its desicion on the fact if it is dealing with I/O or CPU-bound processes. This way, a I/O bound process will be executed earlier than a CPU bound process because a CPU bound process will keep the CPU busy for longer periods of time, while I/O bound processes have smaller CPU bursts so they can be executed fairly fast while not having to wait for a CPU bound process.

5.2

a) When we want to have a high CPU utilization, we need the CPU to be utilised all the time, but when we also want to have a high response time, we may prefer the shorter CPU burst cycles over processes that have a longer CPU burst. This is contraversial because switching processes means that the CPU utilization will go down, because the CPU is not doing anything between the switches.
b) When the waiting time of a process goes up, the turnaround time goes up as well. So when we want to have maximum waiting time, we can't have an avarage turnaround time because the turnaround time count the waiting time plus execution and I/O operations.
c) This is not conflicting with each other because the CPU bursts of processes do not depend on I/O bursts of processes, the follow up eachother. When scheduling right, the CPU and I/O utilization will be high.

5.5

a) This way, the formula only uses its latest prediction to calculate this prediction. The prediction will equal 100 ms.
b) This way, the formula uses the latest known burst time of the process for 99% in its equation. The other 1% is made up of the last prediction. The prediction will equal t + 0,1 ms.

5.7

a) This exercise is at the end of this document.

b)	FCFS:	(2 + 3 + 11 + 15 + 20) / 5	=	10,2
	SJF:	(1 + 3 + 7 + 12 + 20) / 5	=	8,6
	Non-preemptive:	(8 + 13 + 17 + 19 + 20) / 5	=	15,4
	RR:	(2 + 3 + 20 + 13 + 18) / 5	=	11,2
c)	FCFS:	(0 + 2 + 3 + 11 + 15) / 5	=	6,2
	SJF:	(0 + 1 + 3 + 7 + 12) / 5	=	4,6
	Non-preemptive:	(0 + 8 + 13 + 17 + 19) / 5	=	11,4
	RR:	(0 + 2 + 12 + 9 + 13) / 5	=	7,2

d) Algorithm SJF results in the minimal average waiting time.

5.10

Shortest Job First and Priority scheduling.

5.12

a) When the time quantum is 1 millisecond, the RR scheduler will give each process 1 millisecond of CPU time before switching to another one. All I/O bound processes will execute for one millisecond and will then be send to the I/O waiting state, they will each do I/O operations for 10 milliseconds, so when the last one finishes executing the first I/O process will be ready in the queue. The CPU bound process will only have 1 millisecond to execute before all other I/O processes will have the CPU for another 10 milliseconds (plus overhead) in total. This way, the CPU bound process will get little time executing.
b) Now, the time quantum is 10 milliseconds, but each I/O process will still only execute for 1 millisecond because after that millisecond they will need to execute I/O operations. This way the CPU bound process will get 10 milliseconds of running time before being interrupted by all the other I/O processes.

6.2

1. Mutual exclusion is preserved.
   This is the case because no two processes can enter their critical sections at the same time. For one process to enter the critical section, the other one needs to not be in their critical section.
2. The progress requirement is satisfied.
   The two processes each execute after each other so this requirement is met as well.
3. The bounded-waiting requirement is met.
   The two processes each execute after each other so this requirement is also met.