HW5

1.Operating System Concept Chapter 5 Exercises: 5.3 5.4 5.5 5.8

5.3

process	arrival time	burst time
P1	0	8
P2	0.4	4
P3	1	1

```
avg turnaround time=[Σ(Pi's waiting time + Pi's burst time)]/n
```

- a. FCFS scheduling algorithm?

```
avg turnaround time=[(0 + 8) + (7.6 + 4) + (11 + 1)] /3 = 10.53
```

- b. SJF scheduling algorithm?

```
avg turnaround time=[(0 + 8) + (8.6 + 4) + (7 + 1)] /3 = 9.53
```

- c. future-knowledge scheduling.

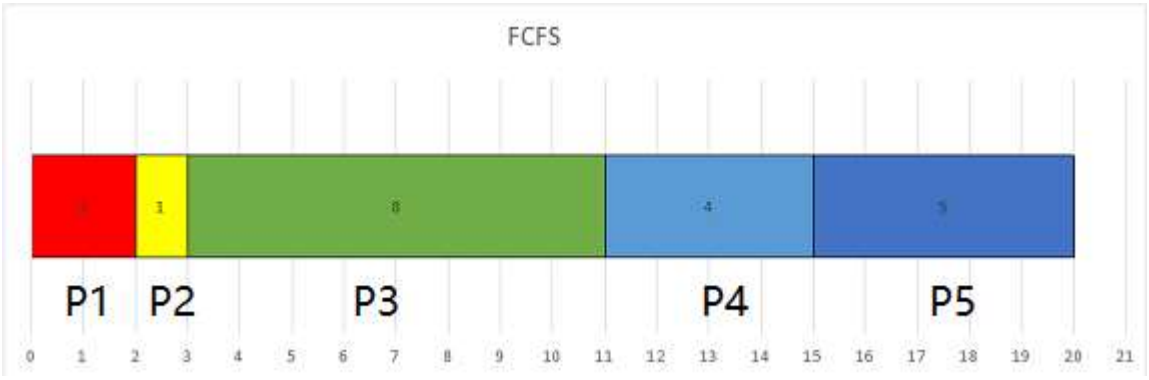
```
avg turnaround time=[(6 + 8) + (1.6 + 4) + (0 + 1)] /3 = 6.86
```

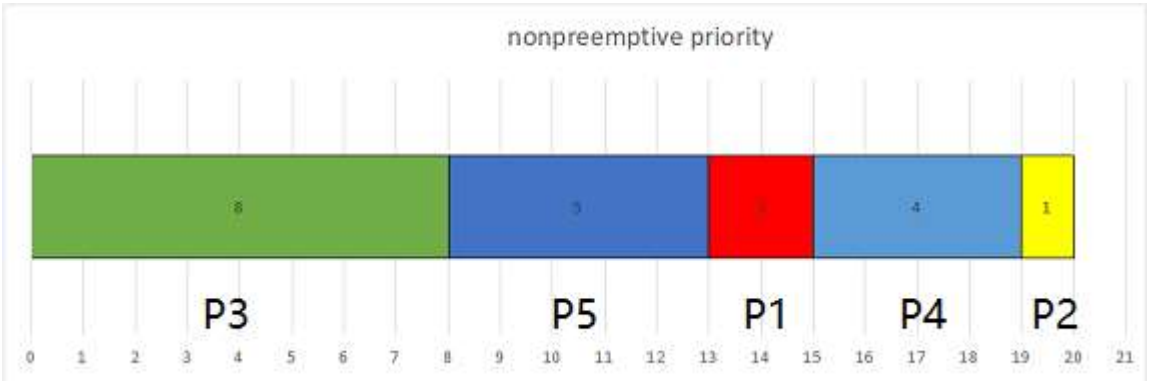
5.4

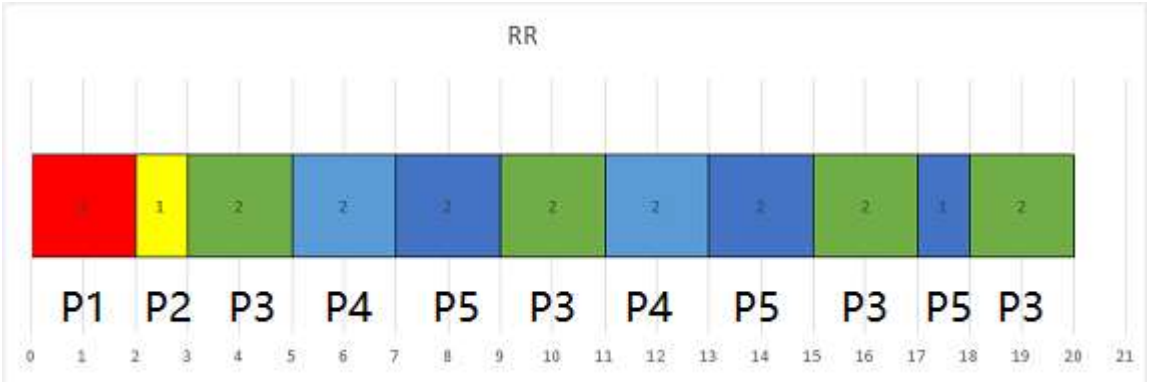
process	burst time	priority
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	5	3

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

- a. Draw four Gantt charts.







- b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

process	FCFS	SJF	nonpreemptive priority	RR
P1	2	3	15	2
P2	3	1	20	3
P3	11	20	8	20
P4	15	7	19	13
P5	20	12	13	18
avg	10.2	8.6	15	11.2

- c. What is the waiting time of each process for each of these scheduling algorithms?

process	FCFS	SJF	nonpreemptive priority	RR
P1	0	1	13	0
P2	2	0	19	2
P3	3	12	0	12
P4	11	3	15	9
P5	15	7	8	13
avg	6.2	4.6	11	7.2

- d. Which of the algorithms results in the minimum average waiting time (over all processes)?

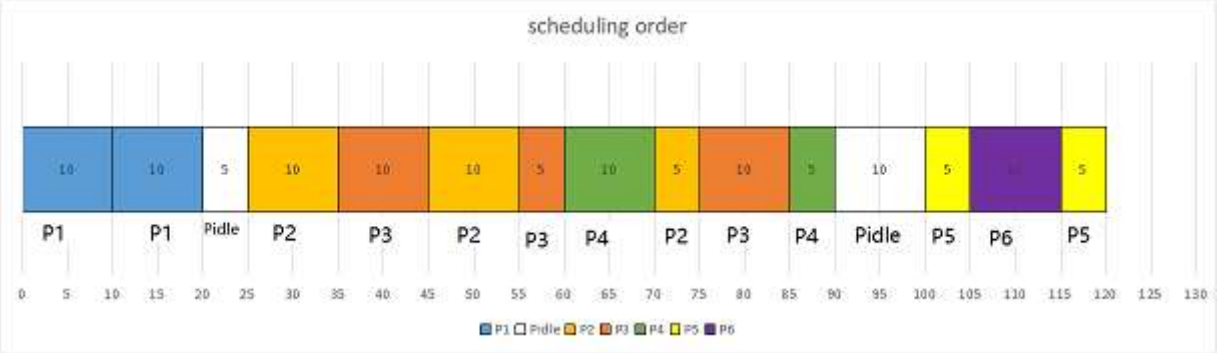
  “

  SJF

5.5

process	priority	burst time	arrival
P1	40	20	0
P2	30	25	25
P3	30	25	30
P4	35	15	60
P5	5	10	100
P6	10	10	105

- a. Show the scheduling order of the processes using a Gantt chart.



- b. What is the turnaround time for each process?
- c. What is the waiting time for each process?

```
The answers of the above two questions(b,c) are both in the table bellow:
```

process	turnaround time	waiting time
P1	20	0
P2	50	25
P3	5	30
P4	30	15
P5	20	10
P6	10	0

- d. What is the CPU utilization rate?

```
CPU utilization rate=105/120 x 100%=87.5%
```

5.8 Suppose that a CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

```
It will favor the I/O-bound programs because of the relatively short CPU burst request by them; however, the CPU-bound programs will not starve because the I/O-bound programs will relinquish the CPU relatively often to do their I/O.
```

2. The reason why the Mars Pathfinder had reset itself, and how to avoid that.

- Why:

```
A priority inversion occurred because of the absense of priority inheritance. A watchdog timer noticed that the task with higher priority had not been executed for some time, concluded that something had gone drastically wrong, and initiated a total system reset.
```

- How to avoid:

```
Priority-inheritance protocol must be deployed, so that the low-priority thread would have inherited the priority of the high-priority thread blocked on it while it held the mutex, causing it be scheduled with higher priority than the medium-priority communications task, thus preventing the priority inversion.
```