OS Solution

1. (a) [3 marks for proiduce process, if it is correct, otherwise 0. 5 marks for consumer(s) process, if it is correct, otherwise 0]

Consumer Process(s): Only access the item from the infinite buffer, but do not produce any item.
Producer Process: Can put/access the item into/from the infinite buffer.
Problem:
(1) We have allowed multiple Consumer Processes at the same time.
(2) Only one Producer can access the shared data at the same time
Shared Data Set:

- Semaphore **mutex** initialized to 1
- Semaphore **wrt** initialized to 1
- Integer **count** initialized to 0 (indicates how many CP in the CS)

Producer

```
do
{
wait (wrt) ;     // Putting an item
signal (wrt) ;
}
while (TRUE);
```

Consumer(s)

```
do
{
wait (mutex) ;
count ++ ;
if (count == 1)
{
wait (wrt) ;
}
signal (mutex) // Accessing from buffer
wait (mutex) ;
count-- ;
if (count  == 0)
{
signal (wrt);
}
signal (mutex) ;
}
while (TRUE);
```

1. (b) [2 marks will be given if the answer is correct, otherwise 0]

X, Y, Z are semaphores initialized to 1

P_1	P_2	P_3
wait (X); wait (Y); wait (Z); . . . signal (X); signal (Y); signal (Z);	wait (Y); wait (Z); wait (X); . . . signal (Y); signal (X); signal (Z);	wait (Z); wait (X); wait (Y); . . . signal (Z); signal (X); signal (Y);

Q 3. or Q. 3(a)

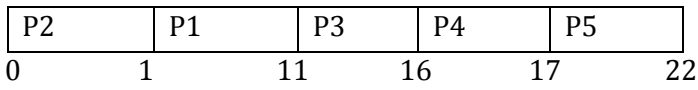
Fair-Share Scheduling

Group	G1	G2	G3
Process	P1	P2	P3
Weightage	0.5	0.25	0.25
Base Priority	60	30	30

Time	P1			P2			P3			Who executes
	Base Priority	Process CPU count	Group CPU count	Base Priority	Process CPU count	Group CPU count	Base Priority	Process CPU count	Group CPU count	
T=0	60	0	0	**30**	**0 . . . 60**	**0 . . . 60**	30	0	0	**P2**
T=1	60	0	0	75	30	30	**30**	**0 . . . 60**	**0 . . . 60**	**P3**
T=2	60	0	0	**52**	**15 . . . 75**	**15 . . . 75**	75	30	30	**P2**
T=3	60	0	0	85	37	37	**52**	**15 . . . 75**	**15 . . . 75**	**P3**
T=4	60	0	0	**57**	**18 . . . 78**	**18 . . . 78**	85	37	57	**P2**
T=5	60	0	0	88	39	39	**57**	**18 . . . 78**	**18 . . . 78**	**P3**
	60	0	0	**58**	**19**	**19**	88	39	39	**P2**

1 mark each for each row from T=0 to T=5. So total 6 marks.

2.A FCFS Gantt Chart



Marks:
1 : If correct
0 : If Wrong

Process	Burst Time	Arrival	TAT	**NTAT**	Waiting Time
P1	10	1	10	**1**	0
P2	1	0	1	**1**	0
P3	5	2	14	**2.8**	9
P4	1	3	14	**14**	13
P5	5	4	18	**3.6**	13

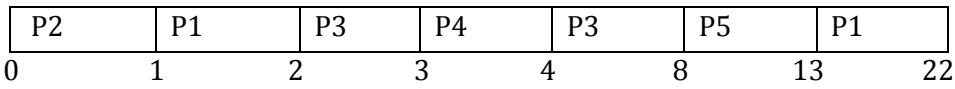
Marks:
1 : If correct
0 : If Wrong

*NTAT = Normalized Turnaround Time

Average Waiting Time = (0+0+9+13+13)/5 = 35/5 **= 7**

Marks:
1 : If correct
0 : If Wrong

2.B SRTF Gantt Chart



Marks:
1 : If correct
0 : If Wrong

Process	Burst Time	Arrival	TAT	**NTAT**	Waiting Time
P1	10	1	21	**2.1**	11
P2	1	0	1	**1**	0
P3	5	2	6	**1.2**	1
P4	1	3	1	**1**	0
P5	5	4	9	**1.8**	4

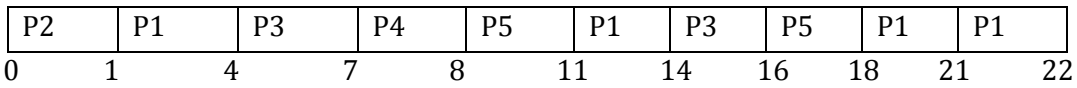
*NTAT = Normalized Turnaround Time

Marks:
1 : If correct
0 : If Wrong

Average Waiting Time = (11+0+1+0+4)/5 = 16/5 **= 3.2**

Marks:
1 : If correct
0 : If Wrong

2.C Round Robin (RR) Gantt Chart with Quantum of 3 Units



Marks:
1 : If correct
0 : If Wrong

Process	Burst Time	Arrival	TAT	**NTAT**	Waiting Time
P1	10	1	21	**2.1**	11
P2	1	0	1	**1**	0
P3	5	2	14	**2.8**	9
P4	1	3	5	**5**	4
P5	5	4	14	**2.8**	9

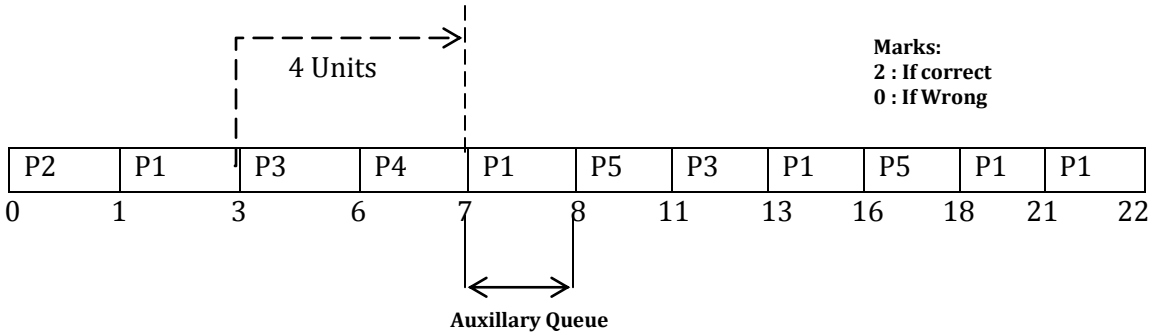
*NTAT = Normalized Turnaround Time

Marks:
1 : If correct
0 : If Wrong

Average Waiting Time = (11+0+9+4+9)/5 = 33/5 **= 6.6**

Marks:
1 : If correct
0 : If Wrong

2.D Virtual Round Robin (RR) Gantt Chart with Quantum of 3 Units



Marks:
2 : If correct
0 : If Wrong

Auxillary Queue = Basic Time Quantum of the Process - (Time Spent by the Process in Running before it was selected from the Ready Queue)

Auxillary Queue = 3-2 = 1 Unit

4. T4 ---------- Undo/Roll Back

T2,T3 --------- Redo

T1 ---------- No Need to Recover

Marks:
2 : If correct
0 : If Wrong

5(A) Need Matrix = Max Claim – Allocation

Need Matrix =

	R1	R2	R3	R4
P1	0	0	0	0
P2	0	7	5	0
P3	6	6	2	2
P4	2	0	0	2
P5	0	3	2	0

(1 Mark)

5(B)

P1 can complete the execution first (Need (P1) < Available)
Available = Current Available + Allocation to Process (P1)
Available = [2 1 1 2]

P4 can complete the execution first (Need (P4) < Available)
Available = Current Available + Allocation to Process (P4)
Available = [4 4 6 6]

P5 can complete the execution first (Need (P5) < Available)
Available = Current Available + Allocation to Process (P5)
Available = [4 7 9 8]

P2 can complete the execution first (Need (P2) < Available)
Available = Current Available + Allocation to Process (P2)
Available = [6 7 9 8]

P3 can complete the execution first (Need (P3) < Available)
Available = Current Available + Allocation to Process (P1)
Available = [6 7 12 12]

Safe Sequence is :<P1 P4 P5 P2 P3> **(3 Marks)**
Yes, the system is in Safe State

5(C) Assume, it can be granted

Current Available = [2 0 0 0]

Need (P3) = [6 5 2 2]

Allocation (P3) = [0 1 3 4]

Now, P1 can complete the execution first (Need (P1) < Available)

Available = Current Available + Allocation to Process (P1)
Available = [2 0 1 2]

P4 can complete the execution first (Need (P4) < Available)
Available = Current Available + Allocation to Process (P4)
Available = [4 3 6 6]

P5 can complete the execution first (Need (P5) < Available)
Available = Current Available + Allocation to Process (P5)
Available = [4 6 9 8]

Now, P2 can't complete its execution as it requires 7 Units of R2 whereas only 6 Units are available

Similarly, P3 can't complete its execution as it requires 6 Units of R1 whereas only 4 Units are available

Hence, P2 & P3 are in deadlock **(3 Marks)**