KIIT Deemed to be University
Online Mid Semester Examination(Spring Semester-2021)

Subject Name & Code: OS **Applicable to Courses:B.Tech**

Full Marks=20 **Time:1 Hour**

SECTION-A(Answer All Questions. All questions carry 2 Marks)
Time:20 Minutes **(5×2=10 Marks)**

<table>
<tr><th>Question No</th><th>Question Type(MCQ/SAT)</th><th>Question</th><th>Answer Key(if MCQ)</th></tr>
<tr><td>Q.No:1(a)</td><td>MCQ</td><td>Which of the following operation is performed when an interrupt occurs.
(a) Interrupt is ignored
(b) The change in the state of the interrupted process after processing of interrupt
(c) Save the state of the interrupted process and reload the state of the new process
(d) Resume the interrupted process’s execution after processing of interrupt</td><td>C</td></tr>
<tr><td></td><td>MCQ</td><td>Which of the following information are saved when a context switch is occurred among the processes.
(a) Translation look-aside buffer
(b) Program Counter
(c) General purpose registers
(d) None of the above</td><td>B and C</td></tr>
<tr><td></td><td>MCQ</td><td>Which of the following below statements are correct with respect to user level and kernel level threads
(a) Context switch is faster in kernel threads
(b) A system call in user thread can block the entire process
(c) User threads are transparent to kernel
(d) All of the above</td><td>B and C</td></tr>
<tr><td></td><td>MCQ</td><td>Consider three processes P1, P2, and P3 with burst time 10, 20, and 30 units and the arrival times are 0, 2, and 6 respectively. Determine the number of context switches (without considering the context switches at start and end) are required, if OS uses SRTF scheduling algorithm.
(a) 3
(b) 4
(c) 2
(d) 1</td><td>C</td></tr>
<tr><td>Q.No:1(b)</td><td>MCQ</td><td>Consider three processes P1, P2 and P3 along with their burst time and arrival time are specified in the below table.
<table>
<tr><th>Processes</th><th>Burst Time (ms)</th><th>Arrival Time(ms)</th></tr>
<tr><td>P1</td><td>9</td><td>0</td></tr>
<tr><td>P2</td><td>4</td><td>1</td></tr>
<tr><td>P3</td><td>9</td><td>2</td></tr>
</table></td><td>C</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Find the average waiting time (ms) of above three processes using preemptive shortest job first scheduling algorithm.
(a) 4.33
(b) 7.33
(c) 5.0
(d) 6.33</td><td></td></tr>
<tr><td></td><td><u>MCQ</u></td><td>Consider three processes P0, P1 and P2 along with their burst time and arrival time are specified in the below table.
<table>
<tr><td>Processes</td><td>Burst Time (ms)</td><td>Arrival Time(ms)</td></tr>
<tr><td>P0</td><td>5</td><td>0</td></tr>
<tr><td>P1</td><td>7</td><td>1</td></tr>
<tr><td>P2</td><td>4</td><td>3</td></tr>
</table>
Determine the order of completion of above three processes using Round Robin scheduling algorithm with time quantum of 2 units.
(a) P0, P2, and P1
(b) P0, P1, and P2
(c) P1, P0, and P2
(d) P2, P1, and P0</td><td>A</td></tr>
<tr><td></td><td><u>MCQ</u></td><td>Consider five processes P0, P1, P2, P3 and P4 along with their burst time and arrival time are specified in the below table.
<table>
<tr><td>Processes</td><td>Burst Time (ms)</td><td>Arrival Time(ms)</td></tr>
<tr><td>P0</td><td>6</td><td>0</td></tr>
<tr><td>P1</td><td>2</td><td>3</td></tr>
<tr><td>P2</td><td>4</td><td>5</td></tr>
<tr><td>P3</td><td>6</td><td>7</td></tr>
<tr><td>P4</td><td>3</td><td>10</td></tr>
</table>
Find the average turnaround time (ms) of above five processes using shortest remaining time first scheduling algorithm.
(a) 7.2
(b) 6
(c) 8.5
(d) 5.6</td><td>A</td></tr>
<tr><td></td><td><u>MCQ</u></td><td>Consider five processes P0, P1, P2, P3 and P4 along with their burst time, arrival time, priority (lower number is the higher priority) are specified in the below table.
<table>
<tr><td>Processes</td><td>Burst Time (ms)</td><td>Arrival Time(ms)</td><td>Priority</td></tr>
<tr><td>P0</td><td>11</td><td>0</td><td>2</td></tr>
<tr><td>P1</td><td>28</td><td>5</td><td>0</td></tr>
<tr><td>P2</td><td>2</td><td>12</td><td>3</td></tr>
<tr><td>P3</td><td>10</td><td>2</td><td>1</td></tr>
<tr><td>P4</td><td>16</td><td>9</td><td>4</td></tr>
</table>
Find the average waiting time (ms) of all the processes using preemptive priority scheduling algorithm.
(a) 28
(b) 29
(c) 32
(d) 26</td><td>B</td></tr>
<tr><td><u>Q.No:1(c)</u></td><td><u>MCQ</u></td><td>Consider the following statements.

1-FCFS scheduling by its nature is always Non-Preemptive type.
2-A process can directly go from waiting state to Running State</td><td>C</td></tr>
</table>

		3-Round Robin Scheduling is best suited for interactive Processes 4-Middle Term Scheduler is required when there is absence of long term Scheduler or presence of Long term scheduler with minimal functionality. 5-Switching between threads of same process is inefficient and takes more time as compared to process switching. 6-Starvation is a problem with Fixed Priority Scheduling. 7-SJF Scheduling results into optimal average waiting time of the processes. Which of the above mentioned statements are true? (a)1,2,3,4 & 7 only (b)1,3,4,5 & 6 only (c)1,3,4,6 & 7 only (d)1,2,4,5 & 6 only	
	MCQ	Consider the following statements. 1-FCFS scheduling by its nature is always Preemptive type. 2-A process can directly go from Running state to Ready State 3-Multilevel feedback queue scheduling has simpler algorithm as compared to multilevel queue scheduling. 4-Switching between threads of same process is efficient and takes less time as compared to process switching. 5-Starvation problem can be resolved with aging priority. 6-SJF Scheduling results into large average waiting time of the processes. 7-In Round Robin Scheduling a process is bound to get the CPU attention after a fixed time. Which of the above mentioned statements are false? (a)1,2,4 & 6 only (b)1,3 & 6 only (c)2,3,6 & 7 only (d)1,3, & 7 only	B
	MCQ	Consider the following statements. 1-A Short Term Scheduler is in charge of waiting to ready state transition of a process. 2-A process will not suffer from starvation in Round Robin Scheduling. 3-Switching between threads of different processes is same as process switching. 4-A single program can give rise to several processes 5-The response time and waiting time of the processes are same for FCFS scheduling. 6-Dispatch latency is a part of the context switching time 7-When processes arrive at the same time, the Preemptive and Non Preemptive version of priority scheduling will result into different waiting time of the processes. Which of the above mentioned statements are true?	C

		(a)1,2,3,4 & 7 only (b)1,3,4,5 & 6 only (c)2,3,4,5 & 6 only (d)2,4,5,6 & 7 only	
	MCQ	Consider the following statements. 1-A Short Term Scheduler is in charge of waiting to ready state transition of a process. 2-SJF Scheduling results into optimal average waiting time of the processes. 3-Switching between threads of different processes is same as process switching. 4-Middle Term Scheduler is required when there is absence of long term Scheduler or presence of Long term scheduler with minimal functionality. 5-The response time and waiting time of the processes are same for FCFS scheduling. 6-Dispatch latency is a part of the context switching time 7-Process Control Block has a Pointer field to add them to the particular scheduling queue implemented as a linked list Which of the above mentioned statements are true? (a)2,3,4,5,6 & 7 only (b)1,3,4,5 & 6 only (c)2,3,4,5 & 6 only (d)2,4,5,6 & 7 only	A

Q.No:1(d)	**MCQ**	In a uniprocessor system, there are two concurrent processes A and B as follows:
		
		Process A:
		Begin
		WAIT(S);
		Read(x);
		Increase x by 1;
		SIGNAL(S);
		End
		
		Process B:
		Begin
		DI; //Disable Interrupt
		Read(x);
		Left shift x once;
		EI; //Enable Interrupt
		End
		
		Semaphore S=1;
		unsigned int x =3; // x is a global variable
		
		What would be the smallest value of x from all possible values of x after completion A and B?
		(A) 4
		(B) 5
		(C) 6
		(D) 7
			A

	MCQ	In a uniprocessor system, there are two concurrent processes A and B as follows:	D

		<table><tr><td>Process A:</td><td>Process B:</td></tr><tr><td>Begin
		WAIT(S);
		Read(x);
		Increase x by 1;
		SIGNAL(S);
		End</td><td>Begin
		DI; //Disable Interrupt
		Read(x);
		Left shift x once;
		EI; //Enable Interrupt
		End</td></tr></table>
		Semaphore S=1;
		unsigned int x =3; // x is a global variable
		
		What would be the sum of all possible values of x after completion A and B?
		(A) 21
		(B) 25
		(C) 15
		(D) None of the above	

	MCQ	In a uniprocessor system, there are two concurrent processes A and B as follows:
		
		<table><tr><td>Process A:</td><td>Process B:</td></tr><tr><td>Begin
		WAIT(S);
		Read(x);
		Increase x by 1;
		SIGNAL(S);
		End</td><td>Begin
		DI; //Disable Interrupt
		Read(x);
		Left shift x once;
		EI; //Enable Interrupt
		End</td></tr></table>
		Semaphore S=1;
		unsigned int x =3; // x is a global variable
		
		How many different possible values of x can be there after completion A and B?
		(A) 2
		(B) 3
		(C) 4
		(D) None of the above
			B

	MCQ	In a uniprocessor system, there are two concurrent processes A and B as follows:
		
		<table><tr><td>Process A:</td><td>Process B:</td></tr><tr><td>Begin
		WAIT(S);
		Read(x);
		Decrease x by 1;
		SIGNAL(S);
		End</td><td>Begin
		DI; //Disable Interrupt
		Read(x);
		Left shift x once;
		EI; //Enable Interrupt
		End</td></tr></table>
			A

		Semaphore S=1; unsigned int x =2; // x is a global variable What would be the smallest value of x from all possible values of x after completion A and B? (A) 1 (B) 2 (C) 5 (D) 3	
Q.No:1(e)	**MCQ**	Semaphore mutex=1; Each process Pi, i = 1, 2, …, 9 is coded as follows: P(mutex); { Critical Section } V(mutex); The code for P10 is as follows: for j = 1 to 3 do V(mutex); { Critical Section } V(mutex); What is the largest number of processes that can be inside the critical section at any moment? (a) 1 (b) 3 (c) 7 (d) 10	C
	MCQ	Semaphore mutex=1; Each process Pi, i = 1, 2, …, 8 is coded as follows: P(mutex); { Critical Section } V(mutex); The code for P9 is as follows: for j = 1 to 2 do V(mutex); { Critical Section } V(mutex); What is the largest number of processes that can be inside the critical section at any moment? (a) 1 (b) 3 (c) 5 (d) 9	C
	MCQ	Semaphore mutex=1; Each process Pi, i = 1, 2, …, 9 is coded as follows: P(mutex); { Critical Section } V(mutex);	4

		The code for P10 is as follows: V(mutex); V(mutex); { Critical Section } V(mutex); What is the largest number of processes that can be inside the critical section at any moment? (a) 1 (b) 3 (c) 5 (d) 9	
	MCQ	Semaphore mutex=1; Each process Pi, i = 1, 2, …, 9 is coded as follows: P(mutex); { Critical Section } V(mutex); The code for P10 is as follows: V(mutex); V(mutex); V(mutex); { Critical Section } P(mutex); What is the largest number of processes that can be inside the critical section at any moment? (a) 1 (b) 3 (c) 7 (d) 10	5

SECTION-B(Answer Any One Question. Each Question carries 10 Marks)

Time: 30 Minutes **(1×10=10 Marks)**

Question No	Question
Q.No:2	The Cats and Mice Problem: (10 marks) A number of cats and mice inhabit a house. Your job is to synchronize the cats and mice so that the following requirements for the common food dish are satisfied: If a cat is eating at the food dish, other cats cann't share the food dish at the same time. But, if any mouse is eating at the food dish, other mice can share the food dish. If any cat see the mice/mouse with the food dish, then the cat

must eat the mice/mouse. No mouse should ever get eaten.

Answer the following for the above "The Cats and Mice Problem"

i. List all of the synchronization primitives and shared variables that you have used to synchronize the cats and mice and identify the purpose of each one.

ii. Explain why it is not possible for mice to be eaten by cats under your synchronization technique.

Justify whether it is not possible for cats or mice to starve under your synchronization technique.

Sol

The Cat and Mice problem

Problem:

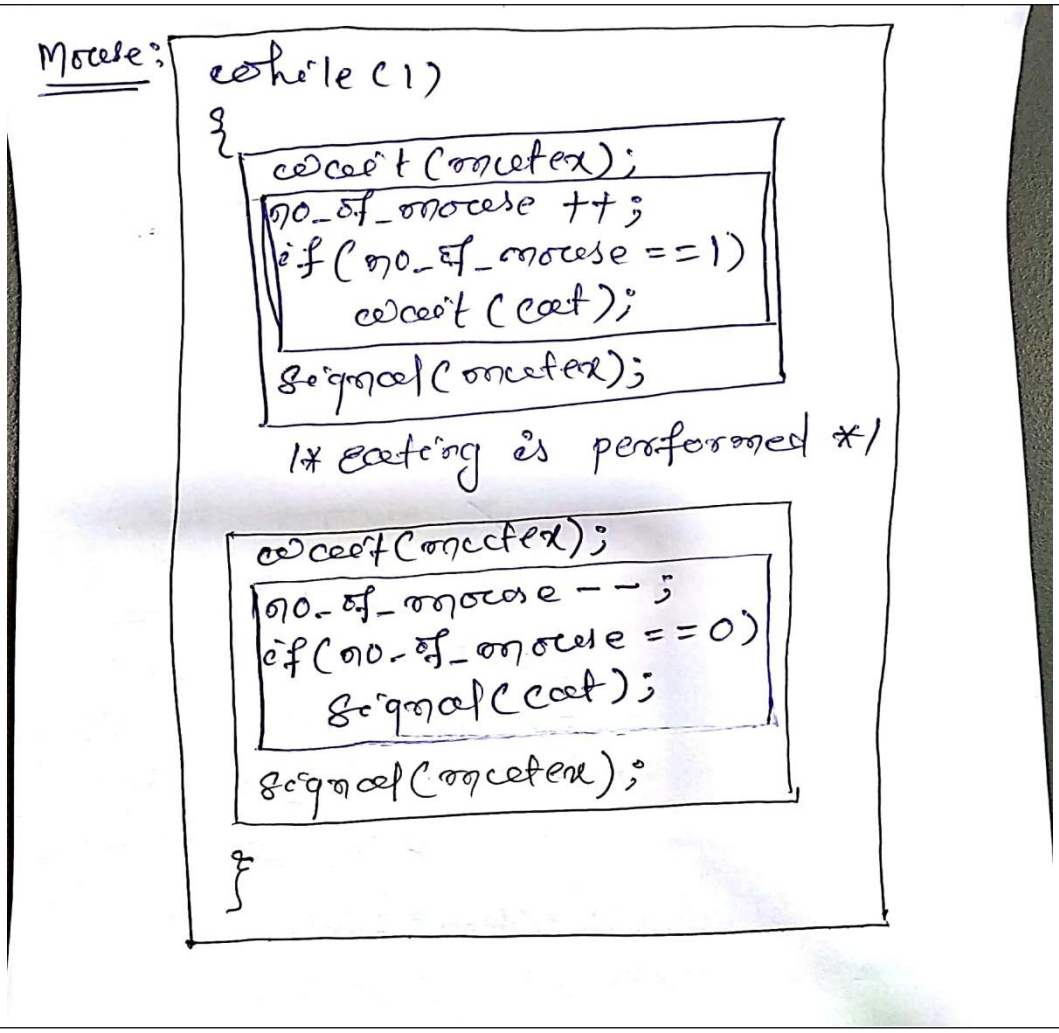
1. Multiple Cats can't eat simultaneously
i.e. $Cat_1, Cat_2, \ldots Cat_n$ ✗
2. Multiple mice can eat simultaneously
i.e. $m_1, m_2, \ldots m_n$ ✓
3. Both Cat and mice are not allowed
i.e. Cat, mouse ✗

Sol^n to the cat and mice problem

```
semaphore cat = 1;
semaphore mutex = 1;
int no_of_mouse = 0;
```

cat:

```
while(1)
{
    wait(cat);
    /* Eating is performed */
    signal(cat);
}
```



Q.No:3

a) Which process states diagram would you propose for system which requires neither any I/O operations nor any other process suspension? (2 marks)

b) Consider process arrival as given below where N = right most significant digit of your Roll No.(ex:- for Roll No. 190854, N=**4**):

Process	CPU Burst Time(ms)	Arrival Time	Priority
A	4	0	2
B	3	6	3
C	6	N	6
D	5	3	N
E	1	4	4

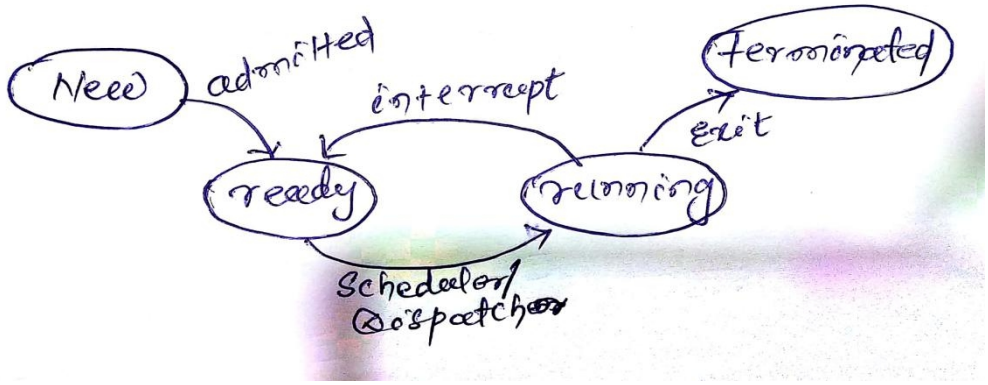
Calculate the following for *priority (non preemptive)* and *round robin* (time quantum = 2 ms) CPU scheduling algorithm:

i. Average waiting time
ii. Turnaround time for each process
iii. Order of completion

(*hints:-higher digits indicate higher priority*) (8marks)

Sol

A)

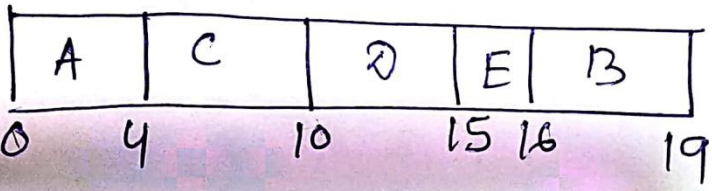


B)

Consider roll no. 190854, i.e. N=4

(i) Non-preemptive priority

Process	BT	AT	Priority	CT	TAT	COT
A	4	0	2	4	4	0
B	3	6	3	19	13	10
C	6	4	6	10	6	0
D	5	3	4	15	12	7
E	1	4	4	16	12	11
						5.6



(i) Avg. COT = 5.6

(ii) TAT(A, B, C, D, E) = (4, 13, 6, 12, 12)

(iii) Order of completion of process are (A, C, D, E, B)

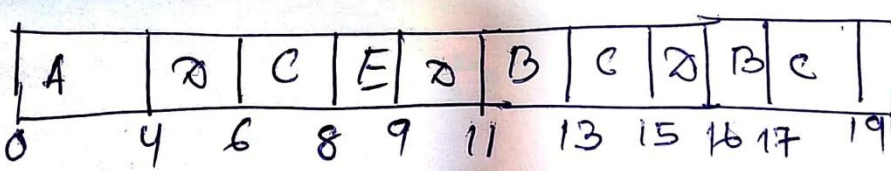
(ii) RR with time quantum = 2 ms

Process	BT	AT	CT	TAT	WT
A	4	0	4	4	0
B	3	6	17	11	8
C	6	4	19	15	9
D	5	3	16	13	8
E	1	4	9	5	4

5.8

Queue: A D C E D B C D B C

Gantt Chart



(i) Avg. WT = 5.8

(ii) TAT (A, B, C, D, E) = (4, 11, 15, 13, 5)

(iii) Order of completing is (A, E, D, B, C)

Q.No:4 Let there is an array of size n, which is used for storing both stack data and heap data. The stack data is stored from the left end of the array, i.e. from 0th index, towards right end of the array, and the heap data is stored from the right end of the array, i.e. from (n-1)th index, towards left end of the array. In this system, there are 4 kinds of processes such as stack producer, heap producer, stack consumer and heap consumer. The stack producers store the data sequentially one after another from the left to right and the heap producers store the data sequentially one after another from the right to left. The stack consumer consumes the last inserted data into the stack and heap consumer consumes the last inserted data into the heap side. The stack producer and stack consumer will be identified through even process id(pid), and the heap producer and heap consumer will be identified through odd process id(pid). Write the synchronization method for producer and consumer that satisfies the following:

- Two producer, one belongs to stack and another belongs to heap, can access the array simultaneously.
- Two consumer, one belongs to stack and another belongs to heap, can access the array simultaneously.
- One producer belongs to stack and another another consumer belongs to heap can access the array simultaneously.

One producer belongs to heap and another another consumer belongs to stack can access the array simultaneously. (10 marks)

Sol

Semaphore empty=n, full=0, mutex_stack=1, mutex_heap=1, count_stack=0, count_heap=0;
int st_i=0,hp_i=n-1;
int arr[n];

Producer	Consumer

p(empty)
if(pid%2){
p(mutex_heap)
arr[hp_i--]=heap_item
v(mutex_heap)
v(count_heap)
}else{
p(mutex_stack)
arr[st_i++]=stack_item
v(mutex_stack)
v(count_stack)
}
v(full)
	if(pid%2){
	p(count_heap)
	p(full)
	p(mutex_heap)
	heap_receive=arr[++hp_i]
	v(mutex_heap)
	}else{
	p(count_stack)
	p(full)
	p(mutex_stack)
	stack_receive=arr[--st_i]
	v(mutex_stack)
	}
	v(empty)

Q.No:5

Consider process arrival as given below where (10 marks)

Z = 11 * (1+ (right most significant digit of your Roll No % 9)). (Ex:- for Roll No. 190560, M=11 *(1+(0%9)) = 11)

Process	CPU Burst Time(ms)	Arrival Time
P_1	30	0
P_2	40	Z
P_3	50	10
P_4	Z	20
P_5	10	30

Calculate the following for *round robin* (time quantum = 15 ms) CPU scheduling algorithm:

a) Average waiting time

b) Turnaround time for each process

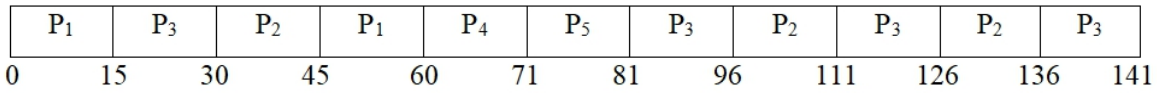
c) Order of completion

Sol

Consider the example:- for Roll No. 190560, M=11 *(1+(0%9)) = 11:

Process	CPU Burst Time(ms)	Arrival Time	Wait Time	TAT	Execution Time
P_1	~~30~~ ~~15~~ 0	~~0~~ ~~15~~ 60	0+30 =30	60	0+15+15+15 + 15+11+10+15
P_2	~~40~~ ~~25~~ ~~10~~ 0	Z=~~11~~ ~~45~~ ~~111~~ 136	19+51+15=85	125	

P_3	~~50~~ ~~35~~ ~~20~~ ~~5~~ 0	~~10~~ ~~30~~ ~~96~~ ~~126~~ **141**	5+51+15+10=81	131	+15+15+10+5 =141
P_4	Z=~~11~~ 0	~~20~~ **71**–	40	51	
P_5	~~10~~– 0	~~30~~ **81**	41	51	
			Avg. Wait = 277/5 = 55.4		



Q.No:6

(a) The two processes execution part defined within if and else statement of the following code. Assume that the execution of ***scanf*** statement is delayed for long time. The ***printf*** statement is completed prior to ***scanf*** statement. What will be the process state of child process when it will gets terminated? Give a suitable reason. (2 marks)

```
i=fork();
if(i>0){

    scanf("%d",&j);
    wait(NULL);
}else if(i==0){
    printf("ABC\n");
}
```

(b) Consider the set of 4 processes whose priority(lower value becomes more priority), arrival time, and burst time are given below:

Process No.	**Arrival Time**	**Priority**	**Burst Time**		
			CPU Burst	**I/O Burst**	**CPU Burst**
P1	0	2	3	1	2
p2	1	x	1	4	3
P3	3	3	2	2	1
P4	2	y	2	1	2

Here

y= (Your roll no) % 2

x= ((Your roll no) / 100) % 2

For example, if roll no is **2106653**, then x=0 and y=1.

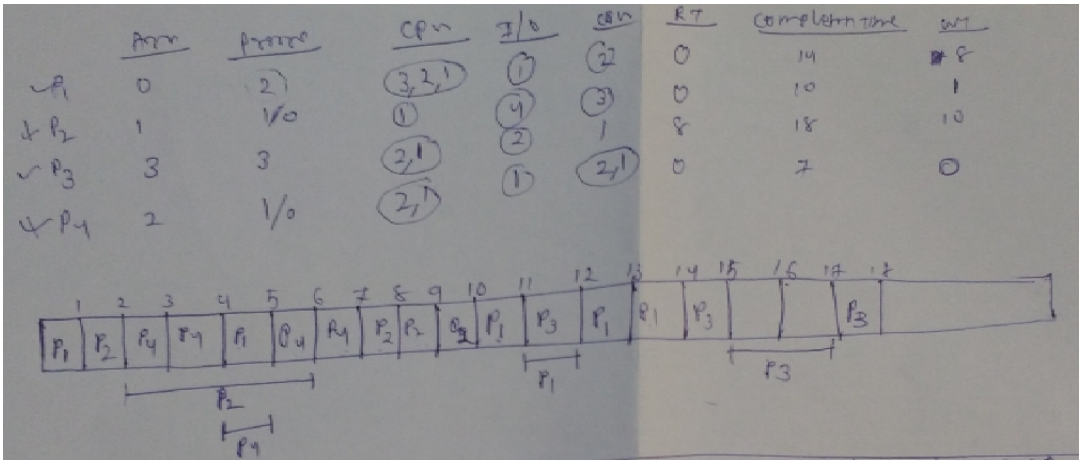
If the OS uses priority based(with preemption) Scheduling, then draw the gantt chart for the whole scheduling and calculate the response time, waiting time, and completion time of each process. (**Lower number means higher priority**) (8 marks)

Sol

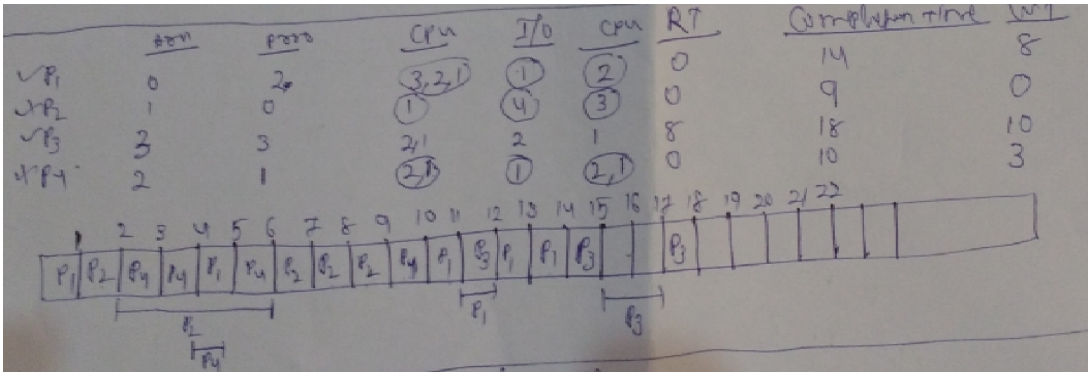
(a) Zombie process

(b)

For (x=0, y=0), (x=1, y=1), (x=1, y=0)



For (x=0, y=1)



Controller of Examinations