Computer Science & Engineering Department, IIT Kharagpur
CS31202 Operating Systems
Class Test 1

Date: 7th Feb, 2025, Time limit: 1 hour Maximum Marks: 20

Roll No: _____ **Name:** _____

Answer all the questions. Answer in the question paper itself.

1. Consider the following two concurrent processes P1 and P2.

```
P1: {
    shared int x;
    x = 10;
    while (1) {
        x = x - 1;
        x = x + 1;
        if (x != 10)
            printf("x = %d", x);
    }
}
```

```
P2: {
    shared int x;
    x = 10;
    while (1) {
        x = x + 1;
        x = x - 1;
        if (x != 10)
            printf("x = %d", x);
    }
}
```

A CPU scheduler of a single-processor system executes these two processes P1 and P2 in a concurrent manner by interleaving their instructions. Show a sequence of execution of P1 and P2 such that statement `"x = 10"` is printed. You should remember that the increment and decrement are not done atomically. **(3)**

	Value of x
P1: x = x – 1	9
P1: x = x + 1	10
P2: x = x + 1	11
P1: if (x != 10)	11
P2: x = x – 1	10
P1: printf("x = %d", x)	10

x = 10 is printed

2. Consider the following solution to the critical section problem for two processes P_0 and P_1. Justify whether the solution ensures mutual exclusion, progress, and bounded waiting.

```
shared int turn = 0;

Process P0
{
        /* Non-critical section */
        turn = 1;
        while (turn == 1);
        /* Critical section */
        turn = 1;
        /* Non-critical section */
}
```

```
Process P1
{
        /* Non-critical section */
        turn = 0;
        while (turn == 0);
        /* Critical section */
        turn = 0;
        /* Non-critical section */
}
```

(3)

Satisfies mutual exclusion and bounded waiting, but not progress. If the two processes run *simultaneously*, one of them will enter its CS based upon the value of done. The other will wait until turn is set in its favor (at the end of the CS of the current holder). If the processes do not run simultaneously, the first process (say P0) sets turn to 1, and loops until P1 arrives and sets turn to 0. Now, P0 enters and exits CS setting turn to 1, so P1 can now enter. In both the cases, mutual exclusion is guaranteed. Bounded wait is also guaranteed, because each process may need to wait only for the other process to access its CS, before it accesses its CS. Progress is not satisfied in the second situation explained above. P0 needs to wait until P1 arrives, that is, even though there is no contention over the CS and P0 is the only process willing to enter its CS, it has to wait for P1 to arrive.

3. Prof. X proposed the following variation of Peterson's solution for **three** processes. However, his proposed solution is incomplete. Complete the solution by filling up the blanks of entry_section(process id) and exit_section(process id).

```
int flag[3] = { 0, 0, 0 };
int turn = 0;

void entry_section (int i) {

    int j = ( i + 1 ) % 3;

    int k = ( i + 2 ) % 3;

    flag[i] = 1 ;

    turn = j ;

    while ( ( flag[j] || flag[k] ) && (turn != i) );
}

void exit_section(int i)
{

        flag[i] = 0;

}
```

(4)

4. Consider a *multilevel feedback queue scheduling algorithm*, which involves three ready queues Q1, Q2 and Q3. The preemptive priority assigned to these ready queues are as follows Q1 > Q2 > Q3. Hence a process in queue Q3 can execute when there is no process in Q1 and Q2. A process in Q2 executes only if there is no process in Q1. Arrival of a process in queue Q1 suspends a running process scheduled from Q2 or Q3. Similarly, arrival of a process in Q2 suspends a running process from Q3.

The scheduling algorithms associated to these queues are as follows:

Q1: Round Robin with quantum q = 3
Q2: Round Robin with quantum q = 5
Q3: Preemptive Remaining Shortest Time Job First (the process in the ready queue with shortest remaining CPU time will be scheduled first)

All the processes initially arrive at their respective designated queue (such as process P1 arrives at queue Q1, whereas process P3 arrives at queue Q2, as shown in Table 1). Once the time quantum q of the process in Q1 expires, the process immediately gets demoted and joins at the rear of queue Q2. Similarly, once the time quanta of the process in Q2 expires, the process immediately gets demoted, and joins at the rear of the queue Q3.

This scheduling algorithm implements aging to make sure that processes should not starve at Q2 and Q3. For instance, if a process waits in the queue Q3 continuously for 6ms, the process will be immediately upgraded, and it will join at the rear of the queue Q2. Similarly, if a process waits in the queue Q2 continuously for 4ms, the process will be immediately upgraded, and it will join at the rear of Q1. Note that this waiting time indicates the time when the process waits in their respective ready queue only, and does not consider the execution on the CPU. Hence, a process in queue Q1 may directly arrive at Q1, or may arrive from Q2 via a queue upgrade. A process in Q2 may directly arrive at queue Q2, or it may arrive from Q1 via demotion, or it may arrive from Q3 via an upgrade. A process in Q3 may directly arrive at Q3, or it may arrive from queue Q2. If two processes arrive at the ready queue at the same time, the process with lower index enters first in the ready queue.

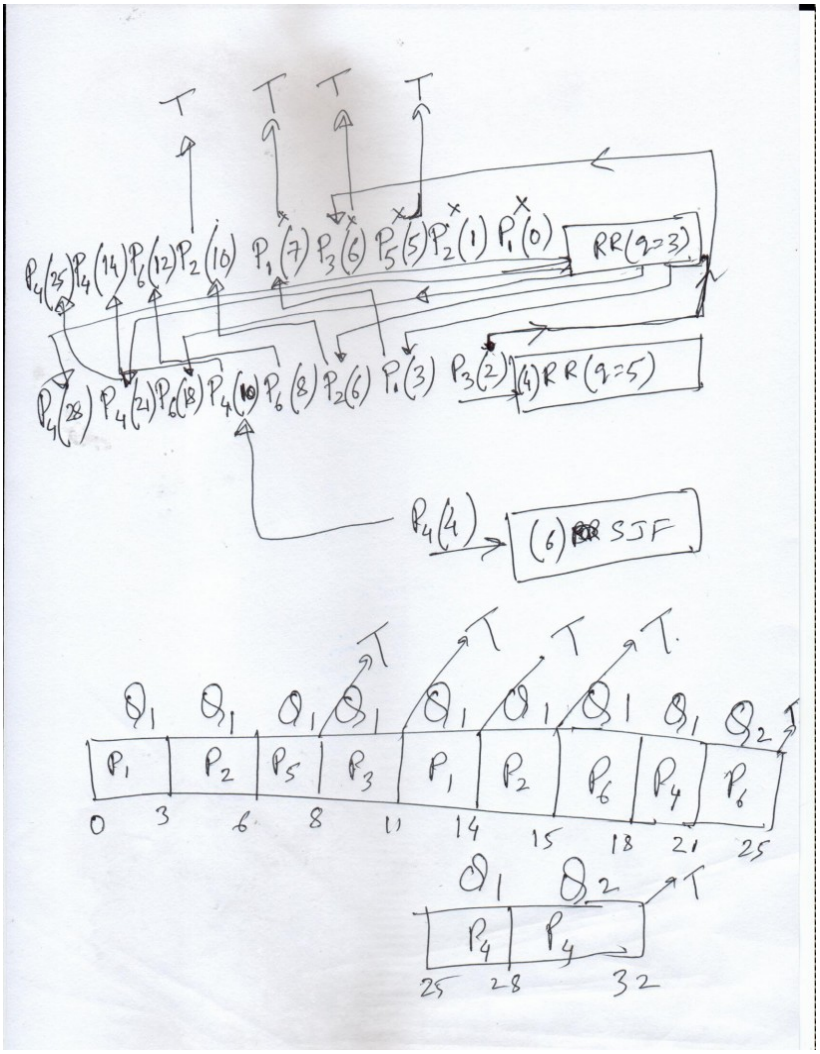
Consider the statistics of six processes below, with their respective arrival time, CPU burst time, and their respective queue of arrival. Draw the (i) Gantt chart and (ii) multilevel feedback queue, and clearly show the arrival of different processes at Q1, Q2 and Q3 (original arrival, via demotion, or via upgrade) with their respective timestamp. Clearly show the upgrade and demotion of each process in the multilevel queue. In the Gantt chart, for each process execution, specify the last ready queue from where the process gets allocated to the CPU. Also mark the termination of each process.

Next, compute the waiting time and the turnaround time of each process, and compute the average waiting and turnaround times.

Process	Arrival time	CPU Burst Time	Queue
P1	0	6	1
P2	1	4	1
P3	2	3	2
P4	4	10	3
P5	5	2	1
P6	8	7	2

Table 1

(10)


T
T
T
T
P4(25) P4(14) P6(12) P2(10) P1(7) P3(6) P5(5) P2(1) P1(0)
RR(q=3)
P4(28) P4(21) P6(18) P4(10) P6(8) P2(6) P1(3) P3(2)
RR(q=5)
P4(4)
(6) SJF
Q1 Q1 Q1 Q1 Q1 Q1 Q1 Q1 Q2
P1 P2 P5 P3 P1 P2 P6 P4 P6
0 3 6 8 11 14 15 18 21 25
Q1 Q2
P4 P4
25 28 32