Siddiqua Nayyer
19L-2495

43/50

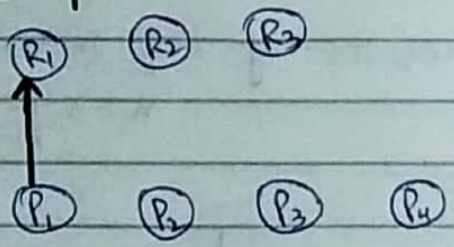
Paper: Advanced Operating Systems
8th July, 2020

10/10

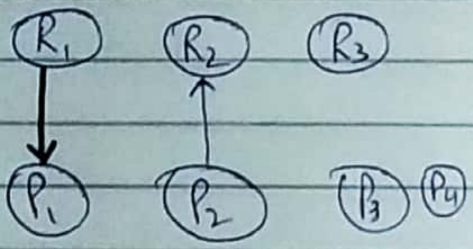
Q#5

Wait and die algorithm
→ Older will wait if resource is not available
→ younger will die in this case.

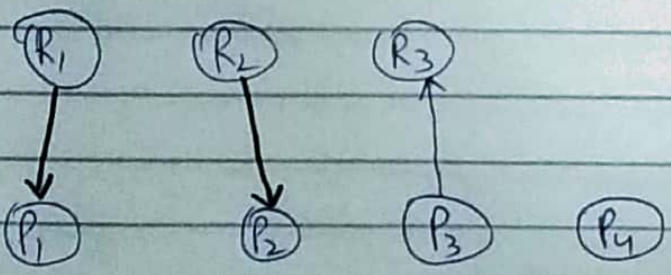
Step 5.1



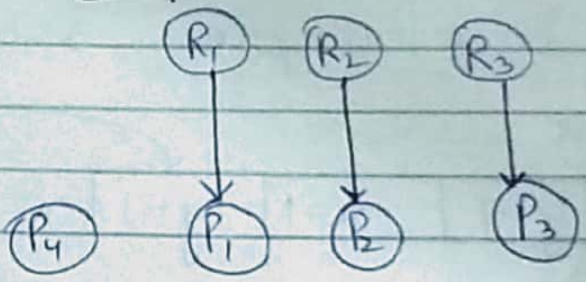
5.2



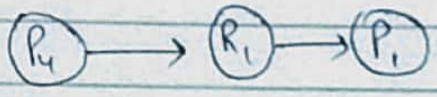
5.3



5.4

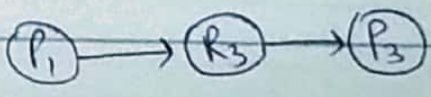


5.5



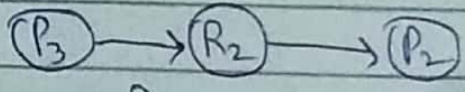
P_1 is old, P_4 dies

5.6



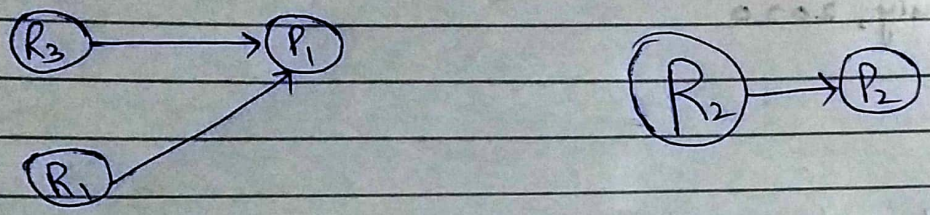
P_1 waits

5.7



P_3 dies

final condition

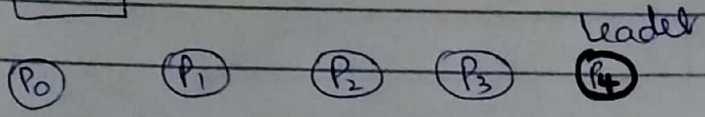


as P_3 dies R_3 is alloted to P_1

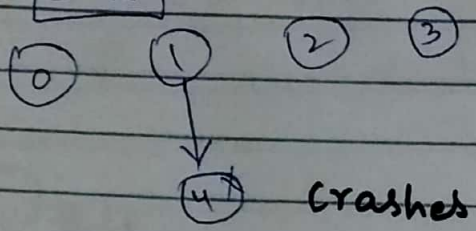
P_4 also died.

Q#3 Bully Algorithm 10/10

3.1

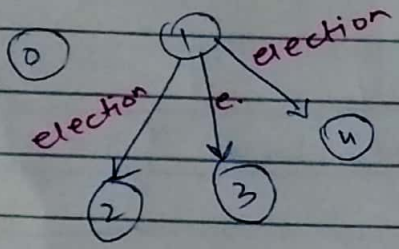


3.2

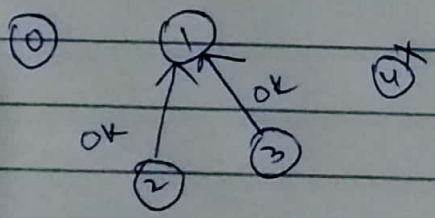


3.3

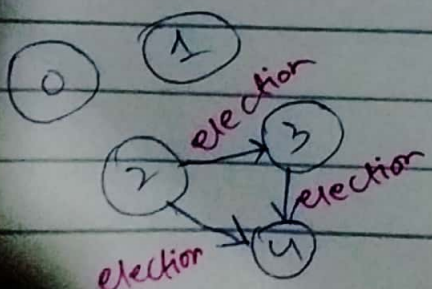
e. => election



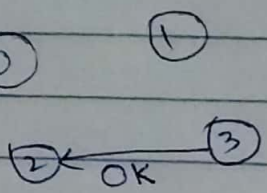
3.4



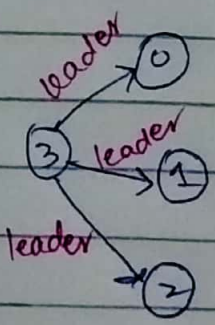
3.5



3.6

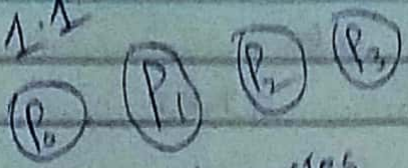


3.7



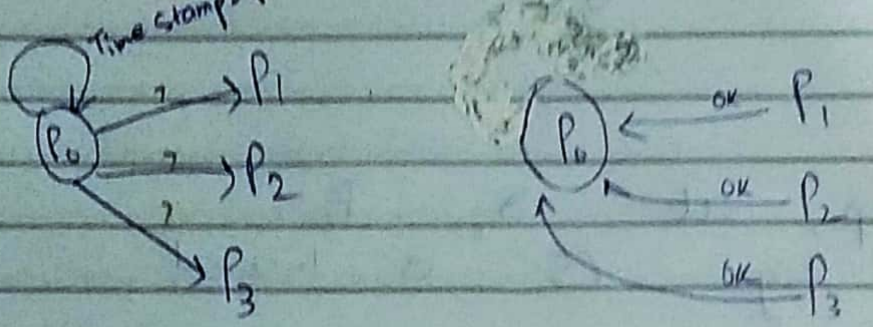
Q#1

1.1



Processes

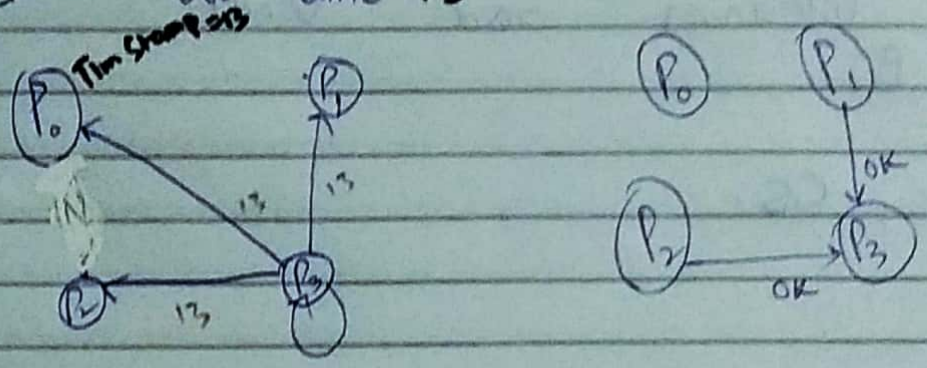
Process P_0 sends msg at 7



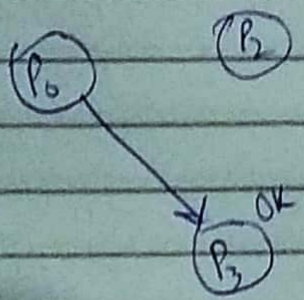
10/10

P_0 ... enter CS

1-2 at time 13



1-3 at time 25



P_3 enter CS

explation

at time 7 P_0 request at CS and sends msg to other process

In next step all say OK, P_0 enters as.

In next step, at 13, P_3 sends request by CS is locked by P_0

at 25, P_0 releases and sends OK to P_3.

P_3 enter CS.

Q#2 Code.

```
RT_Task* EDFMUTT(std::vector<RT_Tasks jobs, int
                                    curr_time)
{
// jobs paramter contains actual task control blocks
    RT_Task* rt = NULL;
// find highest priority job with least deadline
    for (int i=0; i< jobs.size(); i++)
        if ((jobs[i].arrival_time<=curtime) &&
            job[i].execution_time<job[i].end_deadline))
        // see if the task wasn't
        if (rt!=NULL)
        {
        //Compare priority of current job with
                                    previous one if any
            if (jobs[i].priority<=rt->priority)
            {
            // find remaing execution time of current
                                    & previous jobs
            int curr_job_remainingtime = jobs[i].end_deadline
                                    - jobs[i].execution_time;
            int prev_job_remaining = rt->end_deadline -
                                    rt->execution_time
// see if end deadline of current job is less than previous
        if (curr_job_remaing< prev_job_remaining)
            rt = &jobs[i];
        }
    }
return rt; } end function
```

3/10

WRONG: execution time should be added

needless

wrong measure chosen

Q#4

10/10

$y=1$ $x=1$

Given that we cannot shuffle the steps of a single transaction,

This is correct

It is NOT POSSIBLE to write any unique reorder transaction because if in any case, last instruction of T_1 is executed after 1st instruction of T_2, The value of y will become 3, the ~~Correct~~ given sequence of question shows y should be 1 at the end.

BUT If we can shuffle steps

$y=1$

$x=1$

$y=2+y$ $\quad\quad$ $y=1$

$y=1$

$x=1$

$x=2+x$ $\quad\quad$ $x=3$

This is needless, what you figured out was correct....

These three are possible but Illegal

① $y=1$ $\quad$ $y=1$ $\quad$ $x=1$ $\quad$ $x=1$

$y=2+y$ $\quad$ $x=2+x$

② $y=1$ $\quad$ $x=1$ $\quad$ $y=1$ $\quad$ $x=1$

$y=2+y$ $\quad$ $x=2+x$

③ $y=1$ $\quad$ $x=1$ $\quad$ $y=1$ $\quad$ $y=2+y$

$x=1$ $\quad$ $x=2+x$

ANS These are legal but steps are shuffled.

① $y=1$ $\quad$ $x=1$, $x=1$ $\quad$ $y=2+y$

$y=1$ $\quad$ $x=x+2$

② $y=1$, $x=1$, $x=1$, $y=2+y$

$x=x+2$, $y=1$

③ $y=1$, $y=2+y$, $x=1$, $x=1$ $y=1$, $x=x+2$