Name :- Sumit Kumar Yadav

Roll No. :- 18CS30042

Answer 2 :-

(2.1) S0 = 1 , S1 = 0 , S2 = 0

first process P0 executed

wait(S0) → decrease value of S0 ⇒ S0 = 0

print '0' → 1st time

Signal (S1) → increase S1 = 1

Signal (S2) → increase S2 = 1

now process P1, wait (S1) → decrease S1 to 0

signal (S0) → increase S0 to 1

now again process P0 executed

& print '0' → 2nd time

now process P2

wait (S2) → decrease S2 to 0

signal (S0) → increas S0 to 1

again process P0 executed as S0 = 1

so again print '0' → 3rd times.

∴ Process P0 print '0' ⇒ 3 times.

(2.2) n = 0

S = 1

The 2 processes may land up in deadlock.

Consider a case, when consumer performs the wait (S) first and then is waiting on wait (n) as the buffer is empty. Now the producer process

Starts and tries wait (S), and also gets blocked.
Both the processes shall be waiting indefidnitely
resulting in deadlock.

(2.3) ~~One way~~ we have to ensure no deadlock occurs. so for that consider this situation:-

for both the processes, entry section will be

wait (SX); wait (SY); or

wait (SY); wait (SX); both in the same order.

Also for the exit section, for both the processes will be signal (SX); and signal (SY);

in any order.

ie; for process P1:

```
while (true) do {
    wait (SX);
    wait (SY);      } entry section

    X = X+10;
    Y = Y-20;

    signal (SX);
    signal (SY);    } exit section.

}
```

same for process P2.

Ans:- (3.2) 4 processes, 5 allocated resources :-

	Allocated	Maximum	Remaining	order of execution
Process A:	1 0 2 1 1	1 1 2 1 3	0 1 0 0 2	
Process B:	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0	(iii)
Process C:	1 1 0 1 0	2 1 3 1 0	1 0 3 0 0	(ii)
Process D:	1 1 1 1 0	1 1 2 2 1	0 0 1 1 1	(i)

first we calculate remaining for all processes ie (maximum - allocated)

now available : 0 0 x 1 1

it only satisfy process D & $x \geq 1$

now process D terminate & now available resources are

1 1 $(1+x)$ 2 1

it only satisfy process C & $1+x \geq 3$

now process C terminate & now available resources are

2 2 $(1+x)$ 3 1

it only satisfy process B & $1+x \geq 1$ which is already satisfied as above.

now process B terminate & now available resources are

4 2 $(2+x)$ 4 1

it will execute process A as 5^{th} resources must be ≥ 2 but here it is only 1

so, deadlock is always there irrespective of any value of x

∴ for safe state, no value of x exists.

(3.3) $\because$ n processes &
m resources units

$\because$ $\sum_{i=1}^{n} Max_i < m+n$ —(i)

also, $Max_i \geq 1$ for all i —(ii)

because, $need_i = max_i - allocation_i$

if there exist a deadlock state then.

$\sum_{i=1}^{n} allocation_i = m$ —(iii)

using (i) we get,

$\sum need_i + \sum allocation_i = \sum max_i < m+n$

now, using (iii) we get,

$\sum need_i + m < m+n$

$\Rightarrow \sum need_i < n$

~~that~~ this means that there exists a process P_i such that $need_i = 0$. Since, $max_i \geq 1$ it follows that P_i has atleast one resource that it can release

Hence, system cannot be in deadlock state.

Ans 1:-

~~(1.1)~~

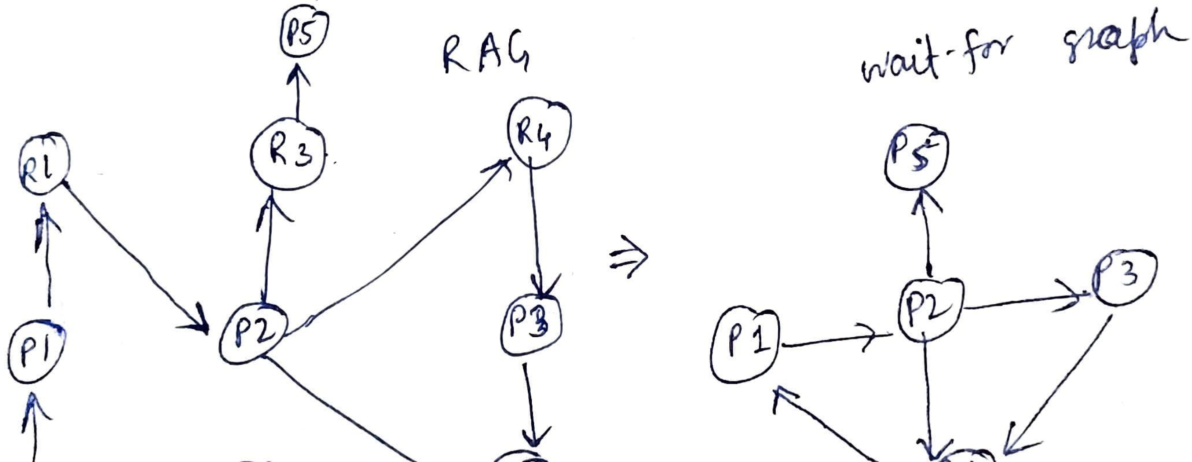
(3.1) for single instance, we use resource-allocation graph.
we periodically run deadlock detection algorith
& if ~~did~~ deadlock detected, we use recovery scheme.

we maintain a wait-for graph
where nodes are processes.

& $P_i \rightarrow P_j$ if P_i is waiting for P_j

& wait-for graph can be derived from resource allocation graph.

ie, consider eg. for converting resource allocation graph into wait-for graph



for deadlock detection:-
→ Periodically invoke an algorithm that seaches for a cycle in wait-for graph
→ if there is a cycle, there exists a deadlock.

Also, detecting a cycle in a graph requires $O(n^2)$ operations, where n is no. of vertices in graph ie. processes.

(4)

(4.1) P_1, P_2, P_3

var = 110

P_1 : var = var − 15

P2 : var = var + 25

P3 : var = var * 3

for maximum :-

order : P2 , P3 , P1

value = 390

for minimum :-

order : P1 , P3 , P2

value = 310

390

135
3
405
15
390

95
3
285
25
310

(4.2) S1, S2, S3

(a) 1621621621621 6 →

S2 = 1 ↑, S1 = ↑ S3 ↑

1 6 2

answer
S2 = 1
S1 = 0
S3 = 0 .

Yes, given strings are possible.

first Thread 3 implement decrease S2 to 0 & print 1 & increase S1 to 1. Now, Thread 2 here decrease S1 to 0 & print 6 and increase S3 to 1. Now, Thread 1 here decrease S3 to 0 print 2 & increase S2 to 1.

Now, same iteration every time print 162162--- in this order.

(b) $S1 = 2$ $\quad S3 = 1$

$S2 = 6$

$\boxed{11226 22}$

$S2=5$	$S2=4$	$S3=0$	Now here
$S1=3$	$S1=4$	$S2=5$	S3 is already zero
$S3=1$	$S3=1$	$S1=4$	

So this is not possible.

first thread 3 (print 1) → thread 3 (print 1) → thread 1 (print 2) → not passible as $S3 = 0$.

Q1: (1.1) (v) ab

(vi) ba

because preemption occurs in the process

(1.2) (v) ab

(vi) ba

because pthread-t thread will preempt

(1.3) (v) ab

(vi) ba

because pthread-t will preempt.

(1.4) (v) ab

(vi) ba

beause pthead will preempt.