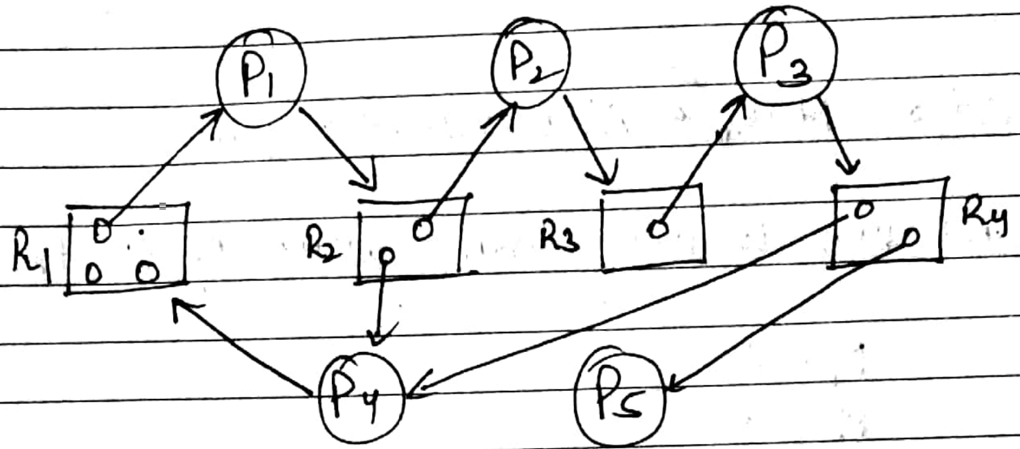
19BCE0215

CAT-II

O. S.

Q1)



a) Allocation matrix :

	R1	R2	R3	R4
P1	1	0	0	0
P2	0	1	0	0
P3	0	0	1	0
P4	0	1	0	1
P5	0	0	0	1

Request matrix :

	R1	R2	R3	R4
P1	0	1	0	0
P2	0	0	1	0
P3	0	0	0	1
P4	1	0	0	0
P5	0	0	0	0

Available matrix :

R_1	R_2	R_3	R_4
2	0	0	0

b) $Need_i$ = Allocation + request

since max is not given, assuming it to be :

$P_1 = 2$ (1,
$P_2 = 2$
$P_3 = 2$
$P_4 = 3$
$P_5 = 1$

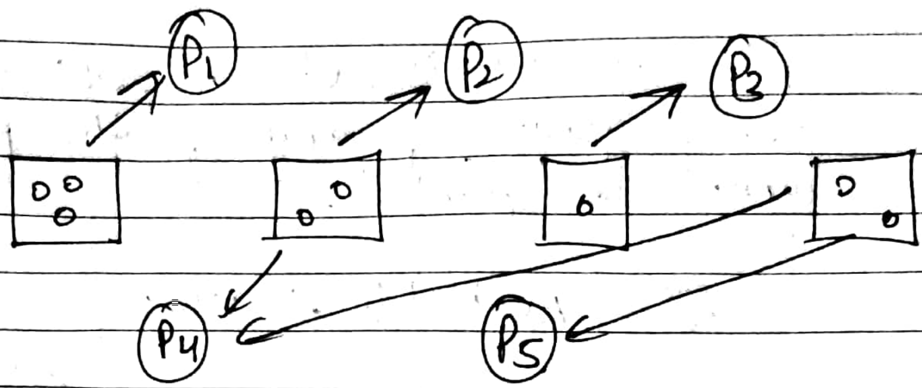
need matrix :

	R_1	R_2	R_3	R_4
P_1	0	1	0	0
P_2				1
P_3				
P_4				
P_5				

∴ need matrix :

	R_1	R_2	R_3	R_4
P_1	1	1	0	0
P_2	0	1	1	0
P_3	0	0	1	1
P_4	1	1	0	1
P_5	0	0	0	1

c)



Resource allocation graph at the current moment.

d) <u>Deadlock detection:</u>

* Deadlock detection algorithm :

1) Let Work and Finish be vectors of length m and n, respectively, initialize :

   (a) Work = Available
   (b) for $i = 1, 2, \ldots n$. if allocation $\neq 0$, then Finish[i] = false, else true.

2) Find an index i such that :

   (a) Finish[i] == false
   (b) $Request_i \leq$ work

   if no such i exists. go to step 4.

3) Work = work + $Allocation_i$
   Finish[i] = true
   go to step 2.

4) If Finish [i] == false, for some i, $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if Finish [i] == false, then P_i is deadlocked.

=> Since there is no safety sequence, Deadlock can be possible.

e)

The system is not in safe state as the requests of any of the processes cannot be granted.

$\therefore$ $\boxed{\text{Available} < \text{need}}$

//

Q3) logical address :

Page no.	offset
8 bits	12 bit

$$\underbrace{\text{Page no.}}_{8 \text{ bits}} \; \underbrace{\text{offset}}_{12 \text{ bit}}$$

main memory size = 256 Kbits

$= 256 \times 2^{10}$

$= 2^8 \times 2^{10}$

$= 2^{18}$ bits

(a) Page size $= 2^x$ $\forall$ $x \rightarrow$ no. of bits of offset

$= \boxed{2^{12}}$ bits.

(b) maximum no. of pages per process.

$= 2^8$

(c) no. of frames $= \dfrac{\text{total memory}}{\text{frame size}}$

$= \dfrac{2^{18}}{2^{12}} = \boxed{2^6}$ frames.

(d) memory size $= 2^{10} \times 2^{10}$

no. of frames $= \dfrac{\text{total memory}}{\text{frame size}} = \dfrac{2^{20}}{2^{12}} = \boxed{2^8}$ frames

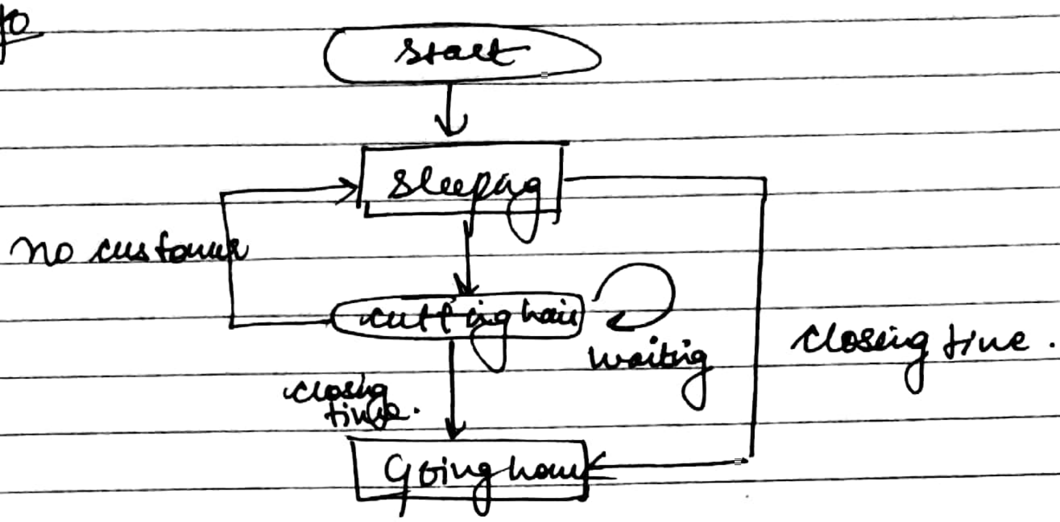
Scanned with CamScanner

Date

Page

Q2

Algo



Semaphore customer = 0
Semaphore Barber = 0
mutex seats = 1;
int free seats = N;

Barber {
while (true)

/waits for customer/.
down (customers);

/ mutex to protect the available seats
down (seats);

free seats ++ ; / gets a free chair

up (Barber); / bring customer for haircut/

Date
Page

cut hair
~~up~~ (seats); /barber is cutting hair/

customer {
while (true) {.
down (seats)
if (free seats > 0)
{
free seats --;
up (customers) /*notify barber*/
up (seats); /*release lock*/
~~down~~ down (barber)
}
else
{
get haircut (seats);
/*customer leaves*/
}
}.

~~hence~~ Hence all the 3 constraints are fulfilled.