

01

Qp code: 39373

Q3 B

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Allocation matrix A

R1	R2	R3
9	3	6

Resource vector R

Consider above snap shot of the system there are four processes (P1-P4) and three resources (R1-R3)

1. Calculate available vector 2. Calculate need matrix 3. Is system in safe state?

Answer

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

C - A

R1	R2	R3
9	3	6

Resource vector R

R1	R2	R3
0	1	1

Available vector V

(a) Initial state

	R1	R2	R3
P1	3	2	2
P2	0	0	0
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	1	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	2	2	2
P2	0	0	0
P3	1	0	3
P4	4	2	0

C - A

R1	R2	R3
9	3	6

Resource vector R

R1	R2	R3
6	2	3

Available vector V

(b) P2 runs to completion

02

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	3	1	4
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	2	1	1
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	1	0	3
P4	4	2	0

C - A

	R1	R2	R3
	9	3	6

Resource vector R

	R1	R2	R3
	7	2	3

Available vector V

(c) P1 runs to completion

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	2

Claim matrix C

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	0	0	2

Allocation matrix A

	R1	R2	R3
P1	0	0	0
P2	0	0	0
P3	0	0	0
P4	4	2	0

C - A

	R1	R2	R3
	9	3	6

Resource vector R

	R1	R2	R3
	9	3	4

Available vector V

(d) P3 runs to completion

P4 runs to completion as all processes are finishing their execution system is in safe state

Q4 B Explain algorithm to avoid deadlock in dining philosopher's problem.

```

/* program diningphilosophers */
semaphore fork[5] = {1};
semaphore room = {4};
int i;
void philosopher (int i)
{
    while (true) {
        think();
        wait (room);
        wait (fork[i]);
        wait (fork [(i+1) mod 5]);
        eat();
        signal (fork [(i+1) mod 5]);
        signal (fork[i]);
        signal (room);
    }
}
void main()
{
    parbegin (philosopher (0), philosopher (1), philosopher (2),
             philosopher (3), philosopher (4));
}

```

Figure 6.13 A Second Solution to the Dining Philosophers Problem