Assignment -01 Deadlock and Deadlock Handing Strategies

Note: Deadline of Submission is 29-mar-2020 and no two students have same set of answers if it found marks will be dedicated .

1) What are the three conditions that must be present for deadlock to be possible?
2) How can hold and wait condition be prevented?
3) List two ways in which the no-preemption condition can be prevented.
4) How can the circular wait condition be prevented?
5) What is the difference among deadlock avoidance, detection, and prevention?
6) Show that four condition of deadlock applied to the given ***figure 1*** below.
7) Show that each of the technique prevention, avoidance and prevention can be apply to the given ***figure 1*** below.

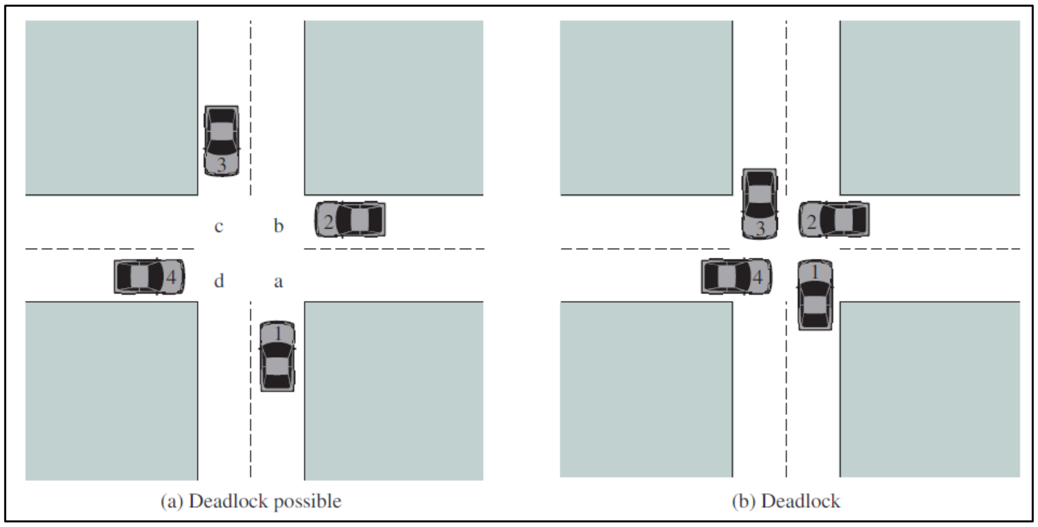


Figure 1

8) Given the following state for the Banker's Algorithm. 6 Processes P0 through P5 , 4 Resource types : A (15 instances); B (6 instances); C (9 instances); D (10 instances) and at time T0 an snapshot is given below in ***figure 3*** then answer the following :
   - Verify that the available array ha been calculated correctly.
   - Calculate the Need matrix.
   - Show that the current state is safe, that is , show sequence of processes.in addition, to sequence show how the Available (working array) changes as each process terminates.
   - Given the request (3,2,3,3) from process P_5 . Should this request be granted? Why or Why not?

9) It was stated that deadlock cannot occur for the situation reflected in ***figure 2***. Justify that statement.

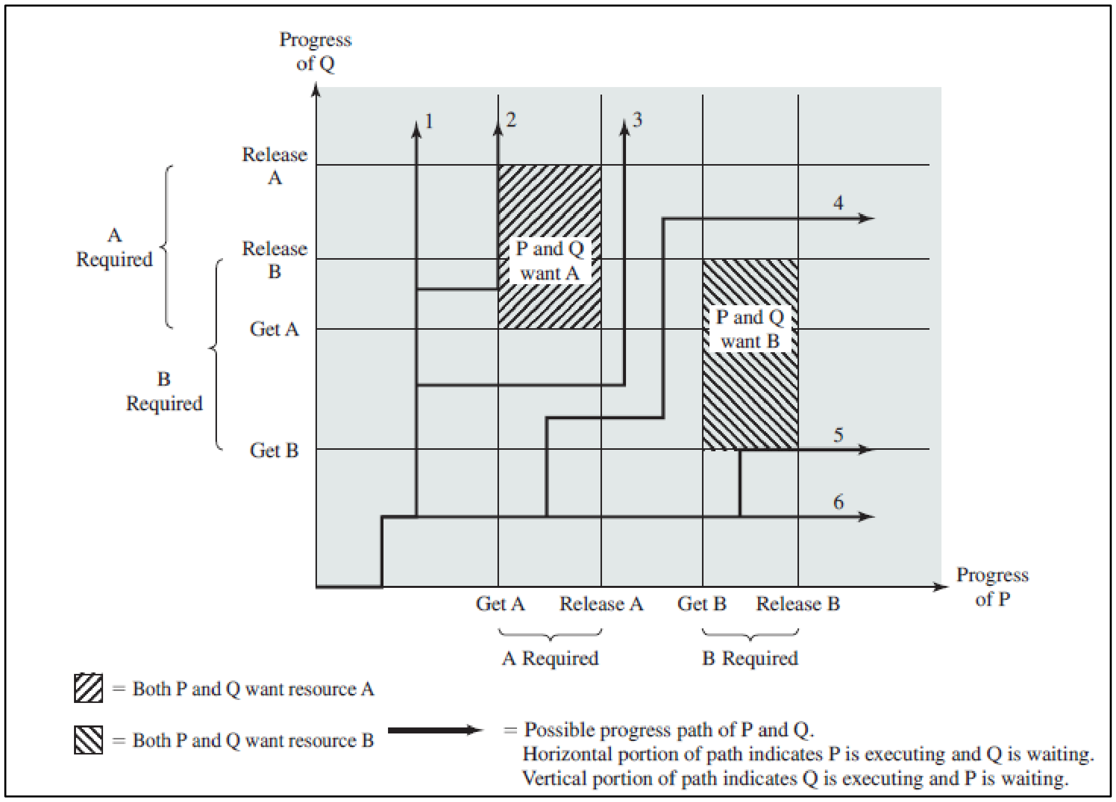
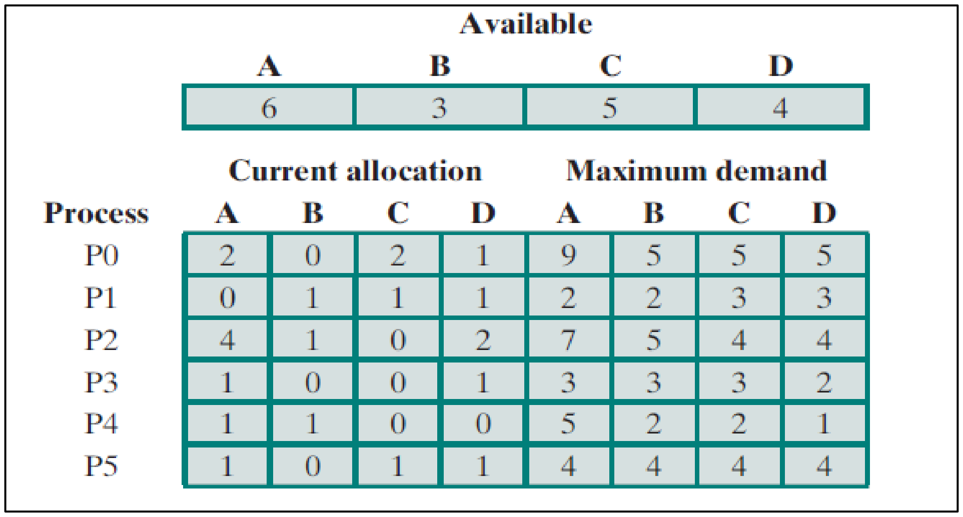

Progress of Q
Release A
Release B
Get A
Get B
A Required
B Required
1
2
3
4
5
6
P and Q want A
P and Q want B
Progress of P
Get A
Release A
Get B
Release B
A Required
B Required
▨ = Both P and Q want resource A
▧ = Both P and Q want resource B
= Possible progress path of P and Q.
Horizontal portion of path indicates P is executing and Q is waiting.
Vertical portion of path indicates Q is executing and P is waiting.

Figure2



Available

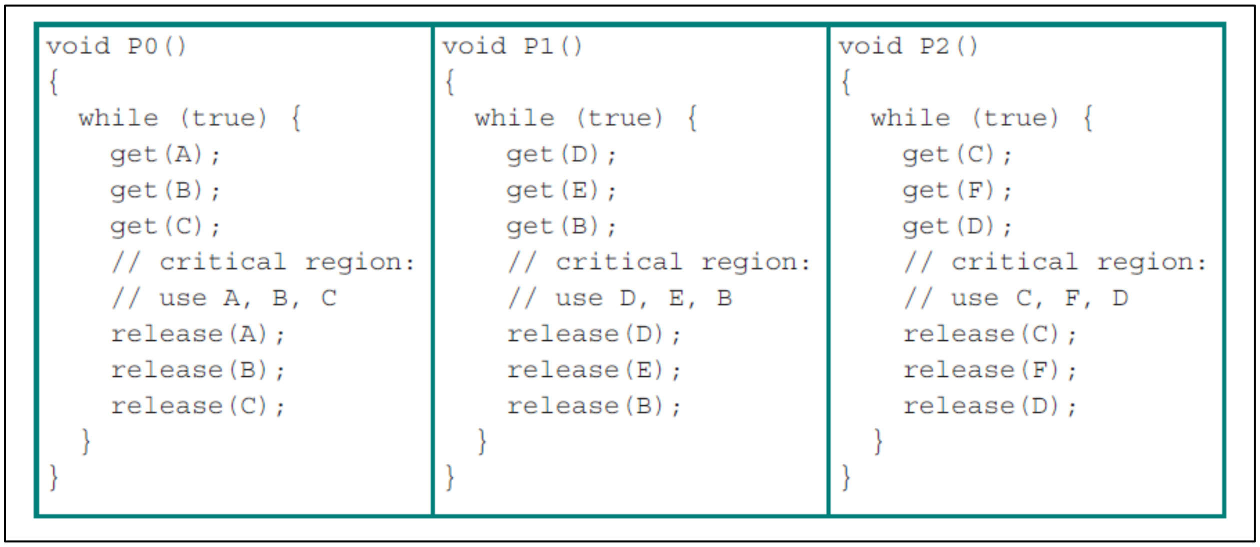
A	B	C	D
6	3	5	4

	Current allocation				**Maximum demand**			
Process	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

Figure3

10) In the code given below in ***figure 4***, three processes are competing for six labeled A to F Then answer the following question:

- Using a resource allocation graph show the possibility of a deadlock in this implementation.
- Modify the order of some of the request to prevent the possibility of any deadlock. You cannot move the requests across procedures, only change the order inside each procedure. Use resource allocation graph to justify your answer.
-



```
void P0()
{
  while (true) {
    get(A);
    get(B);
    get(C);
    // critical region:
    // use A, B, C
    release(A);
    release(B);
    release(C);
  }
}
```

```
void P1()
{
  while (true) {
    get(D);
    get(E);
    get(B);
    // critical region:
    // use D, E, B
    release(D);
    release(E);
    release(B);
  }
}
```

```
void P2()
{
  while (true) {
    get(C);
    get(F);
    get(D);
    // critical region:
    // use C, F, D
    release(C);
    release(F);
    release(D);
  }
}
```

Figure 4

************************* ***Best of Luck*** ****************************