Ans – 1 Round robin

Ans -2 O(mn^2)

Ans 3 n , p

Ans -4 deadlock

Ans -5 8

ans -2 possible

possible

not possible

not possible

possible

Ans 3 Atleast twice

Explanation:

P0 will execute first because only S0=1. Hence it will print 0 (for the first time). Also P0 releases S1 and S2. Since S1=1 and S2=1, therefore P1 or P2, any one of them can be executed.

Let us assume that P1 executes and releases S0 (Now value of S0 = 1). Note that P1 process is completed.

Now S0=1 and S2=1, hence either P0 can execute or P2 can execute. Let us check both the conditions:-

1. Let us assume that P2 executes, and releases S0 and completes its execution. Now P0 executes; S0=0 and prints 0 (i.e. second 0). And then releases S1 and S2. But note that P1 and P2 processes has already finished their execution. Again if P0 tries to execute it goes into sleep condition because S0=0. Therefore, minimum number of times '0' gets printed is 2.

2. Now, let us assume that P0 executes. Hence S0=0, (due to wait(S0)), and it will print 0 (second 0) and releases S1 and S2. Now only P2 can execute, because P1 has already completed its execution and P0 cannot execute because S0 = 0. Now P2 executes and releases S0 (i.e. S0=1) and finishes its execution. Now P0 starts its execution and again prints 0 (thrid 0) and releases S1 and S2 (Note that now S0=0). P1 and P2 has already completed its execution therefore again P1 takes its turn, but since S0=0, it goes into sleep condition. And the processes P1 and P2 which could wakeup P0 has already finished their execution.Therefore, maximum number of times '0' gets printed is 2.

ans 4

9

ans 5

The question is to find the time taken for,

"100 fetch operation and 60 operand read operations and 40 memory

operand write operations"/"total number of instructions".

Total number of instructions= 100+60+40 =200

Time taken for 100 fetch operations(fetch =read)

= 100\*((0.9\*1)+(0.1\*5)) // 1 corresponds to time taken for read

// when there is cache hit

= 140 ns //0.9 is cache hit rate

Time taken for 60 read operations = 60\*((0.9\*1)+(0.1\*5))

= 84ns

Time taken for 40 write operations = 40\*((0.9\*2)+(0.1\*10))

= 112 ns

// Here 2 and 10 the time taken for write when there is cache

// hit and no cahce hit respectively

So,the total time taken for 200 operations is = 140+84+112

= 336ns

Average time taken = time taken per operation = 336/200

ans -6

External fragmentation =0.

Internal fragmentation =160-150) +(1805-1800)+(3180-3176)

= 10 + 5 + 4 =19

Fragmentation = external fragmentation + internal fragmentation = 0 + 19 = 19

ans -7

adv and disdv of continous and index file allocation method

ans -8

0-3(p0)3-8(p1)8-14(p2)14-18(p3)18-21(p4)21-25(p0)25-29(p3)(29-30)p0

ans 09

Given a disk with 100 tracks

And Sequence 45, 20, 90, 10, 50, 60, 80, 25, 70.

Initial position of the R/W head is on track 50.

In SSTF, requests are served as following

Next Served Distance Traveled

50 0

45 5

60 15

70 10

80 10

90 10

25 65

20 5

10 10

-----------------------------------

Total Dist = 130

If Simple SCAN is used, requests are served as following

Next Served Distance Traveled

50 0

60 10

70 10

80 10

90 10

45 65 [disk arm goes to 100, then to 45]

25 20

20 5

10 10

-----------------------------------

Total Dist = 140

Less Distance traveled in SSTF = 130 - 140 = 10

Ans -10 It does not prevent deadlocks, but ensures mutual exclusion.

Explanation:

The code ensures the condition of mutual exclusion: Assume P1 is initiated. It sets wants1=true.  Now since wants2 = false, P1 exists from its while loop and enters its critical section. Now suppose context switch takes place and P2 gets executed. Now it sets wants2=true, and now enters the while loop and remains busy till P1 comes out of the critical section and sets wants1=false, because wants1=true( as set by P1). So we can see that the mutual exclusion condition is satisfied.

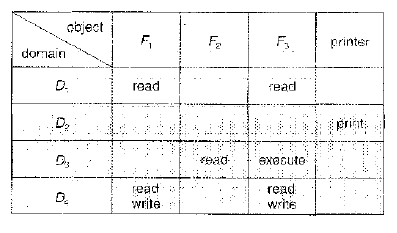
The code does not prevent deadlock: Assume that P1 starts its execution. It sets wants1=true and then gets preempted. Now P2 starts its execution. P2 sets wants2=true and suddenly gets preempted. Now P1 starts execution; it enters the while loop and finds that wants2=true and remains busy in the while loop. Now P1 gets preempted. P2 enters into execution; it enters the while loop and finds that wants1=true remains busy in the while loop. Hence both P1 and P2 remains busy forever.

Ans -11

handwritten attached

Ans -11

a) Our model of protection can be viewed abstractly as a matrix, called an access matrix. The rows of the access matrix represent domains, and the columns represent objects. Each entry in the matrix consists of a set of access rights. Because the column defines objects explicitly, we can omit the object name from the access right. The entry access(/,/) defines the set of operations that a process executing in domain Dj can invoke on object Or To illustrate these concepts, we consider the access matrix shown in Figure There are four domains and four objects—three files (F|, F2, F:1) and one laser printer. A process executing in domain D\ can read files Fj and F3. A process executing in domain D4 has the same privileges as one executing in domain D\; but in addition, it can also write onto files F| and F?. Note that the laser printer can be accessed only by a process executing in domain Do



11 b) Allowing controlled change in the contents of the access-matrix entries requires three additional operations: copy, owner, and control. The ability to copy an access right from one domain (or row) of the access matrix to another is denoted by an asterisk (\*) appended to the access right. The copy right allows the copying of the access right only within the column (that is, for the object) for which the right is defined. For example, a process executing in domain D2 can copy the read operation into any entry associated with file F2. Hence, the access matrix can be modified to the access matrix shown in Figure 14.5(b). This scheme has two variants:

1. A right is copied from access(i, j) to access(k,j); it is then removed from access(i,j). This action is a transfer of a right, rather than a copy.

ANS -12

Handwritten attached

Ans 13

1. **10,13,12,11**

|  |
| --- |
| **Need** |
| **2,0,1,1** |
| **0,6,5,0** |
| **1,1,0,2** |
| **0,0,0,0** |
| **1,4,4,4** |

**© yes**

**Safe sequence – p3 , p4, p0,p1,p2**

**( d)**

**Yes request can be granted immediately**

**No deadlock**

**Safe sequence – p4, p0 , p2 , p3 , p1**