ASSIGNMENT NO: 5

Title : Bankers Algorithm

AIM: Implement C program for Deadlock Avoidance: Banker's Algorithm

THEORY :

- In a multiprogramming environment, several processes may compete for a finite number of resources.
- A process requests resources; if the resources are not available at that time, the process enters a waiting state.
- Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes.
- This situation is called a deadlock. Deadlock avoidance is one of the techniques for handling deadlocks.
- This approach requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime.
- With this additional knowledge, it can decide for each request whether or not the process should wait.
- To decide whether the current request can be satisfied or must be delayed, the system must consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process.
- Banker's algorithm is a deadlock avoidance algorithm that is applicable to a system with multiple instances of each resource type
- It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not.
- Consider there are n account holders in a bank and the sum of the money in all of their accounts is S.
- Every time a loan has to be granted by the bank,it subtracts the **loan amount** from the **total money** the bank has.
- Then it checks if that difference is greater than S.
- It is done because, only then, the bank would have enough money even if all the n account holders drawa ll their moneyat once.

Characteristics of Banker's Algorithm :

- The characteristics of Banker's algorithm are as follows:

1. If any process requests for a resource,then it has to wait.
2. This algorithm consists of advanced features for maximum resource allocation.
3. There are limited resources in the system we have.
4. In this algorithm, if any process gets all the needed resources, then it is that it should return there sources in a restricted period.
5. Various resources are maintained in this algorithm that can fulfill the needs of at least one client.

- **Let us assume that there are n processes and m resource types.**
  - Following data structures are required to implement bankers algorithm for deadlock avoidance.Some data structures that are used to implement the banker's algorithm are:

1. **Available**: It is an array of length 'm' that defines each type of resource available in the system. When Available[j] = K, means that 'K' instances of Resources type R[j] are available in the system.
2. **Max:** It is a [n x m] matrix that indicates each process P[i] can store the maximum number of resources R[j] (each type) in a system.
3. **Allocation:** It is a matrix of m x n orders that indicates the type of resources currently allocated to each process in the system. When Allocation [i, j] = K, it means that process P[i] is currently allocated K instances of Resources type R[j] in the system.
4. **Need:** It is an M x N matrix sequence representing the number of remaining resources for each process. When the Need[i] [j] = k, then process P[i] may require K more instances of resources type Rj to complete the assigned work. Nedd[i][j] = Max[i][j] - Allocation[i][j].
5. **Finish**: It is the vector of the order **m**. It includes a Boolean value (true/false) indicating whether the process has been allocated to the requested resources, and all resources have been released after finishing its task.

Need[i][j] = Max[i][j] - Allocation [i][j]

Example:

Let us consider the following snapshot for understanding the banker's algorithm:

Considering a system with five processes P0 through P4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t0 following snapshot of the system has been taken:

Answer the following questions using the banker's algorithm:

1. What is the reference of the need matrix?
2. Determine if the system is safe or not.
3. What will happen if the resource request (1, 0, 0) for process P1 can the system accept this request immediately?

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Solution:

Ans1:
What will be the content of the Need matrix?

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Need[i]=Max[i]-Allocation[i]
Need for P1: (7, 5, 3) - (0, 1, 0) = 7, 4, 3

Need for P2: (3, 2, 2) - (2, 0, 0) = 1, 2, 2

Need for P3: (9, 0, 2) - (3, 0, 2) = 6, 0, 0

Need for P4: (2, 2, 2) - (2, 1, 1) = 0, 1, 1

Need for P5: (4, 3, 3) - (0, 0, 2) = 4, 3, 1

Apply the Banker's Algorithm:

Available Resources of A, B and C are 3, 3, and 2.

Now we check if each type of resource request is available for each process.
Step 1: For Process P1:
Need <= Available
7, 4, 3 <= 3, 3, 2 condition is **false**.
So, we examine another process, P2.

Step 2: For Process P2:
Need <= Available
1, 2, 2 <= 3, 3, 2 condition **true**
New available = available + Allocation
(3, 3, 2) + (2, 0, 0) => 5, 3, 2
Similarly, we examine another process P3.

Step 3: For Process P3:
P3 Need <= Available
6, 0, 0 < = 5, 3, 2 condition is **false**.
Similarly, we examine another process, P4.

Step 4: For Process P4:
P4 Need <= Available
0, 1, 1 <= 5, 3, 2 condition is **true**
New Available resource = Available + Allocation
5, 3, 2 + 2, 1, 1 => 7, 4, 3
Similarly, we examine another process P5.

Step 5: For Process P5:
P5 Need <= Available
4, 3, 1 <= 7, 4, 3 condition is **true**
New available resource = Available + Allocation
7, 4, 3 + 0, 0, 2 => 7, 4, 5
Now, we again examine each type of resource request for processes P1 and P3.

Step 6: For Process P1:
P1 Need <= Available
7, 4, 3 <= 7, 4, 5 condition is **true**

New Available Resource = Available + Allocation
7, 4, 5 + 0, 1, 0 => 7, 5, 5
So, we examine another process P2.

Step 7: For Process P3:
P3 Need <= Available
6, 0, 0 <= 7, 5, 5 condition is true
New Available Resource = Available + Allocation
7, 5, 5 + 3, 0, 2 => 10, 5, 7

Ans3:
Hence, we execute the banker's algorithm to find the safe state and the safe sequence like P2, P4, P5, P1 and P3.

Ans. 3: For granting the Request (1, 0, 2), first we have to check that **Request <= Available**, that is (1, 0, 2) <= (3, 3, 2), since the condition is true. So the process P1 gets the request immediately.

Conclusion: