EXPERIMENT 9
BANKER'S ALGORITHM FOR DEADLOCK AVOIDANCE

AIM

To implement Banker's Algorithm for deadlock avoidance in a concurrent computing environment.

ALGORITHM FOR OUTPUT()

STEP 0: START

STEP 1: This function is responsible for displaying matrices in a formatted manner

STEP 2: It takes a 2D array as input as input and prints it out, row by row with appropriate column headings.

STEP 3: END

ALGORITHM FOR SAFETY()

STEP 0: START

STEP 1: Initialize variables like-
x - index for safe sequence
flg - flag to check if a process can be executed, target - counter for finished processes

STEP 2: Iterates through each process and checks if its resource needs (N[][]) can be satisfied with available (W[]).

STEP 3: If a process can be executed, it updates the available resources and marks the process as finished.

STEP 4: If all processes can be executed without violating the resource constraints it returns 1 indicating a safe state otherwise, prints a message indicating an unsafe state.

STEP 5: END

ALGORITHM FOR REQUEST()

STEP 0: START

STEP 1: It first checks if the requested resources exceed maximum claim for the process or if the requested resources are currently unavailable.

STEP 2: If the requested resources can be provided, it updates the Allocation Matrix, Need Matrix, and Available vector accordingly.

STEP 3: After resource allocation, it calls the SAFETY() function to ensure that system remains in safe state.

STEP 4: If system is still in safe mode, after the allocation, it prints the safe sequence otherwise, it notifies the user of an unsafe state.

STEP 5: END.

ALGORITHM FOR MAIN

STEP 0: START

STEP 1: Prompts the user to enter the no. of processes (n), the no. of resources (res), the maximum available instance of each resource (R[]), the Allocation Resource Table (A[][]), and Maximum Claim Table (C[][]).

STEP 2: Calculates the Need Matrix (N[][]), which represents the resources still needed by each process to complete its task. This is difference between Maximum claim and Allocated Resources.

STEP 3: Available vector W[] is calculated by finding difference of sum of Allocated Resources from Maximum available resources.

STEP 4: SAFETY() is called to check if system is currently in safe state.

STEP 5: If system is safe, the user is prompted whether to initiate a resource request. If requested, then REQUEST() is called.

STEP 5: END

RESULT

Experiment successfully executed and output obtained.