**EXERCISE-5**

**Banker’s Algorithm** for deadlock Avoidance.

**Aim:**

**To** Implement **Banker’s Algorithm** for deadlock Avoidance.

**Description:**

* A state is safe if the system can allocate resources to each process (up to its maximum)

in some order and still avoid a deadlock. (Or) A system is in a safe state only if there exists a

safe sequence.

Bankers Algorithm for Deadlock AVOIDANCE:

Safe state algorithm

1. Let Work and Finish be vectors of length m and n, respectively. Initialize

Work= Available and Finish[i] =false for i = 0, 1, ... , n - 1.

2. Find an index i such that both

a. Finish[i] ==false

b. Needi ≤ Work

If no such i exists, go to step 4.

3. Work = Work + Allocationi

Finish[i] = true

Go to step 2.

4. If Finish[i] ==true for all i, then the system is in a safe state.

**Program:**

#include<stdio.h>

int main()

{

int pid[10],allocation[10][10],a[5],max[10][10], req[10][10],need[10][10],available[10],finish[10],work[10],safeseq[10],n,m,t=0,i=0,j=0,k=0,noproc=0,rtype=0,flag=0,x=0,tallocation[10][10], tneed[10][10],tavailable[10],tmax[10][10],enter,re,nre,ne,av;

printf("enter number of resource types:");

scanf("%d",&m);

printf("enter the max instances of each resource type:");

for(i=0;i<m;i++)

scanf("%d",&available[i]);

printf("enter number of processes:");

scanf("%d",&n);

printf("enter the max matrix of process:");

for(i=0;i<n;i++)

for(j=0;j<m;j++)

scanf("%d",&max[i][j]);

printf("enter the allocation matrix:");

for(i=0;i<n;i++)

for(j=0;j<m;j++)

{

scanf("%d",&allocation[i][j]);

if(allocation[i][j]<=max[i][j])

x++;

}

for(j=0;j<m;j++)

{

a[j]=0;

for(i=0;i<n;i++)

a[j]=a[j]+allocation[i][j];

}

for(i=0;i<m;i++)

printf("\nallocated instances of %d resource type=%d",i,a[i]);

for(i=0;i<m;i++)

{

if(a[i]<=available[i])

t++;

}

for(i=0;i<m;i++)

available[i]=available[i]-a[i];

for(i=0;i<m;i++)

printf("\navailable instances of %d resource type=%d",i,available[i]);

if(t==m && x==(n\*m))

{

for(i=0;i<n;i++)

{

for(j=0;j<m;j++)

{

need[i][j]=max[i][j]-allocation[i][j];

}

}

for(i=0;i<m;i++)

tavailable[i]=available[i];

for(i=0;i<n;i++)

{

for(j=0;j<m;j++)

{

tneed[i][j]=need[i][j];

tallocation[i][j]=allocation[i][j];

tmax[i][j]=max[i][j];

}

}

do

{

k=0;

for(i=0;i<m;i++)

work[i]=available[i];

for(i=0;i<n;i++)

finish[i]=0;

noproc=n;

while(noproc)

{

for(i=0;i<n;i++)

{

rtype=0;

for(j=0;j<m;j++)

{

if(need[i][j]<=work[j])

rtype++;

}

if(rtype==m&&finish[i]==0)

{

for(j=0;j<m;j++)

work[j]=work[j]+allocation[i][j];

finish[i]=1;

safeseq[k]=i;

k++;

}

}

noproc--;

}

for(i=0;i<n;i++)

{

if(finish[i]==0)

flag=1;

}

if(flag==1)

{

available[re]=tavailable[re];

allocation[nre][re]=tallocation[nre][re];

need[nre][re]=tneed[nre][re];

printf("\nthere is no safe seq for the current resource allocation state");

printf("\nso there is a possibility for deadlock");

}

else

{

printf("\n the safe sequence is");

for(i=0;i<k;i++)

printf("p%d,",safeseq[i]);

printf(".\n,");

}

printf("\nif you want to request the resources for the process press 1:");

scanf("%d",&enter);

if(enter==1)

{

ne=0;

av=0;

printf("enter the process that is requesting :");

scanf("%d",&nre);

printf("enter the resource instances:");

for(j=0;j<m;j++) scanf("%d",&req[nre][j]);

for(j=0;j<m;j++)

{

if(req[nre][j]<=need[nre][j])

ne++;

}

for(j=0;j<m;j++)

{

if(req[nre][j]<=available[j])

av++;

}

if(ne==m)

{

if(av==m)

{

for(i=0;i<m;i++)

tavailable[i]=available[i];

for(i=0;i<n;i++)

{

for(j=0;j<m;j++)

{

tneed[i][j]=need[i][j];

tallocation[i][j]=allocation[i][j];

tmax[i][j]=max[i][j];

}

}

for(j=0;j<m;j++)

{available[j]=available[j]-req[nre][j];

allocation[nre][j]=allocation[nre][j]+req[nre][j];

need[nre][j]=need[nre][j]-req[nre][j]; }

} else{printf("there is no safe sequence, resources are not available and process has to wait "); enter=0; }

}

else{

enter=0;

printf("\nprocess has exceed its maximum limit"); }

}

}while(enter==1);

}

else

{

if(t!=m)

printf("\nallocation is more than available, safe seq is not possible");

else

printf("\nallocation is more than max, safe seq is not possible");

}

}

**Output:**

