Ex.No: 5 IMPLEMENT AN ALGORITHM FOR DEAD LOCK

Date: DETECTION

**AIM:**

To write a C program to implement an Algorithm to detect Deadlock.

**PROBLEM DESCRIPTION:**

Deadlock detection is the process of actually determining that a deadlock exists and identifying the process and resources involved in the deadlock. The basic idea is to check the allocation against resource availability for all possible allocation sequences to determine if the system is in deadlocked state,

The complexity of algorithm is O(N2) where N is the number of proceeds. Another potential problem is starvation; same process killed repeatedly.

**ALGORITHM:**

**1:** Get the no of processes.

**2:** Get the process numbers.

**3:** Get the no of resources types and instances of it.

**4:** Get Max demand of each process of n x m matrices.

**5:** Get the n x m matrices the number of resources of each type currently allocated to each process.

**6:** Calculate the n x m of the remaining resource need of each process.

**7:** Initialize work as available resource and array of finish to false.

**8:** Check the Needed resource is lesser than the available resource if not display the System not in safe state and if it is lesser than system in safe state.

**9:** Initialize work as sum of work and allocation, check if array of finish is true go to step 7 again if not go to step 8.

**10:** Check that request can be immediately granted.

**11:** If single request is lesser than or equal to available if true means arrive to new state.

**12:** Print the sequence if it is in safe state or print not in safe state.

PROGRAM:

#include<stdio.h>

#include<conio.h>

int max[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n,r;

void input();

void show();

void cal();

int main()

{

int i,j;

printf("\*\*\*\*\*\*\*\*\*\* Deadlock Detection Algo \*\*\*\*\*\*\*\*\*\*\*\*\n");

input();

show();

cal();

getch();

return 0;

}

void input()

{

int i,j;

printf("Enter the no of Processes\t");

scanf("%d",&n);

printf("Enter the no of resource instances\t");

scanf("%d",&r);

printf("Enter the Max Matrix\n");

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

{

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

{

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

}

}

printf("Enter the Allocation Matrix\n");

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

{

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

{

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

}

}

printf("Enter the available Resources\n");

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

{

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

}

}

void show()

{

int i,j;

printf("Process\t Allocation\t Max\t Available\t");

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

{

printf("\nP%d\t ",i+1);

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

{

printf("%d ",alloc[i][j]);

}

printf("\t");

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

{

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

}

printf("\t");

if(i==0)

{

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

printf("%d ",avail[j]);

}

}

}

void cal()

{

int finish[100],temp,need[100][100],flag=1,k,c1=0;

int dead[100];

int safe[100];

int i,j;

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

{

finish[i]=0;

}

//find need matrix

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

{

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

{

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

}

}

while(flag)

{

flag=0;

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

{

int c=0;

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

{

if((finish[i]==0)&&(need[i][j]<=avail[j]))

{

c++;

if(c==r)

{

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

{

avail[k]+=alloc[i][j];

finish[i]=1;

flag=1;

}

//printf("\nP%d",i);

if(finish[i]==1)

{

i=n;

}

}

}

}

}

}

j=0;

flag=0;

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

{

if(finish[i]==0)

{

dead[j]=i;

j++;

flag=1;

}

}

if(flag==1)

{

printf("\n\nSystem is in Deadlock and the Deadlock process are\n");

}

else

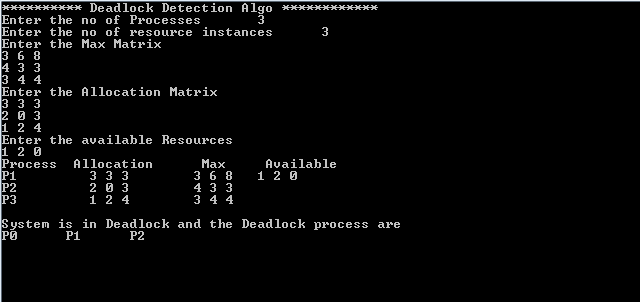
{

printf("\nNo Deadlock Occur");

}

}

SAMPLE INPUT AND OUTPUT:



**CONCLUSION:**

Thus, an algorithm for deadlock detection is implemented using C and executed.