**Aim:** To perform Banker's Algorithm with additional resource request checking to grant the

request or not.

**Algorithm:**

BANKER’S ALGORITHM:

1. Let Work and Finish be vectors of length ‘m’ and ‘n’ respectively.  
   Initialize: Work = Available  
   Finish[i] = false; for i=1, 2, 3, 4….n
2. Find an i such that both  
   a) Finish[i] = false  
   b) Needi <= Work  
   if no such i exists goto step (4)
3. Work = Work + Allocation[i]  
   Finish[i] = true  
   goto step (2)
4. if Finish [i] = true for all i  
   then the system is in a safe state

RESOURCE REQUEST ALGORITHM:

1. If Requesti <= Needi  
   Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2. If Requesti <= Available  
   Goto step (3); otherwise, Pi must wait, since the resources are not available.
3. Have the system pretend to have allocated the requested resources to process Pi by modifying the state as  
   follows:  
   Available = Available – Requesti  
   Allocationi = Allocationi + Requesti  
   Needi = Needi– Requesti

**Code:**

#include<stdio.h>

#include<stdlib.h>

void print(int x[][10],int n,int m){

int i,j;

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

printf("\n");

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

printf("%d\t",x[i][j]);

}

}

}

//Resource Request algorithm

void res\_request(int A[10][10],int N[10][10],int AV[10][10],int pid,int m)

{

int reqmat[1][10];

int i;

printf("\n Enter additional request :- \n");

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

printf(" Request for resource %d : ",i+1);

scanf("%d",&reqmat[0][i]);

}

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

if(reqmat[0][i] > N[pid][i]){

printf("\n Error encountered.\n");

exit(0);

}

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

if(reqmat[0][i] > AV[0][i]){

printf("\n Resources unavailable.\n");

exit(0);

}

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

AV[0][i]-=reqmat[0][i];

A[pid][i]+=reqmat[0][i];

N[pid][i]-=reqmat[0][i];

}

}

//Safety algorithm

int safety(int A[][10],int N[][10],int AV[1][10],int n,int m,int a[]){

int i,j,k,x=0;

int F[10],W[1][10];

int pflag=0,flag=0;

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

F[i]=0;

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

W[0][i]=AV[0][i];

for(k=0;k<n;k++){

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

if(F[i] == 0){

flag=0;

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

if(N[i][j] > W[0][j])

flag=1;

}

if(flag == 0 && F[i] == 0){

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

W[0][j]+=A[i][j];

F[i]=1;

pflag++;

a[x++]=i;

}

}

}

if(pflag == n)

return 1;

}

return 0;

}

//Banker's Algorithm

void accept(int A[][10],int N[][10],int M[10][10],int W[1][10],int \*n,int \*m){

int i,j;

printf("\n Enter total no. of processes : ");

scanf("%d",n);

printf("\n Enter total no. of resources : ");

scanf("%d",m);

for(i=0;i<\*n;i++){

printf("\n Process %d\n",i+1);

for(j=0;j<\*m;j++){

printf(" Allocation for resource %d : ",j+1);

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

printf(" Maximum for resource %d : ",j+1);

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

}

}

printf("\n Available resources : \n");

for(i=0;i<\*m;i++){

printf(" Resource %d : ",i+1);

scanf("%d",&W[0][i]);

}

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

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

N[i][j]=M[i][j]-A[i][j];

printf("\n Allocation Matrix");

print(A,\*n,\*m);

printf("\n Maximum Requirement Matrix");

print(M,\*n,\*m);

printf("\n Need Matrix");

print(N,\*n,\*m);

}

int banker(int A[][10],int N[][10],int W[1][10],int n,int m){

int j,i,a[10];

j=safety(A,N,W,n,m,a);

if(j != 0 ){

printf("\n\n");

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

printf(" P%d ",a[i]);

printf("\n A safety sequence has been detected.\n");

return 1;

}else{

printf("\n Deadlock has occured.\n");

return 0;

}

}

int main(){

int ret;

int A[10][10];

int M[10][10];

int N[10][10];

int W[1][10];

int n,m,pid,ch;

printf("\n DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n");

accept(A,N,M,W,&n,&m);

ret=banker(A,N,W,n,m);

if(ret !=0 ){

printf("\n Do you want make an additional request ? (1=Yes|0=No)");

scanf("%d",&ch);

if(ch == 1){

printf("\n Enter process no. : ");

scanf("%d",&pid);

res\_request(A,N,W,pid-1,m);

ret=banker(A,N,W,n,m);

if(ret == 0 )

exit(0);

}

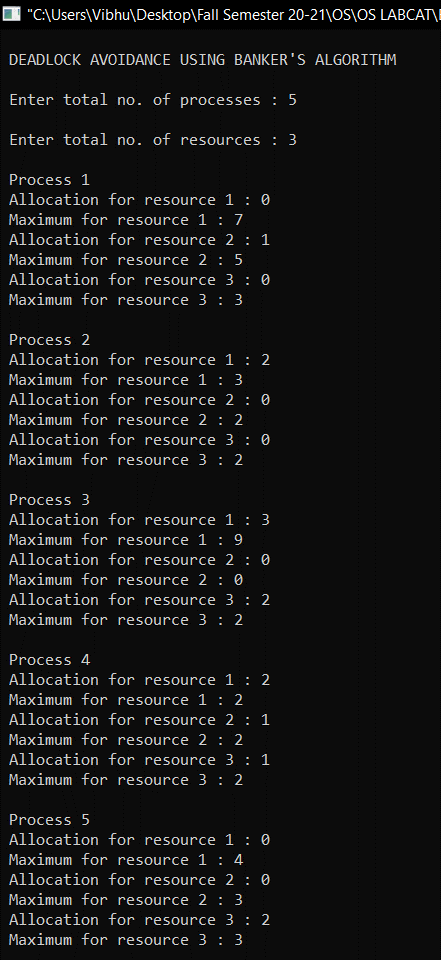
}else

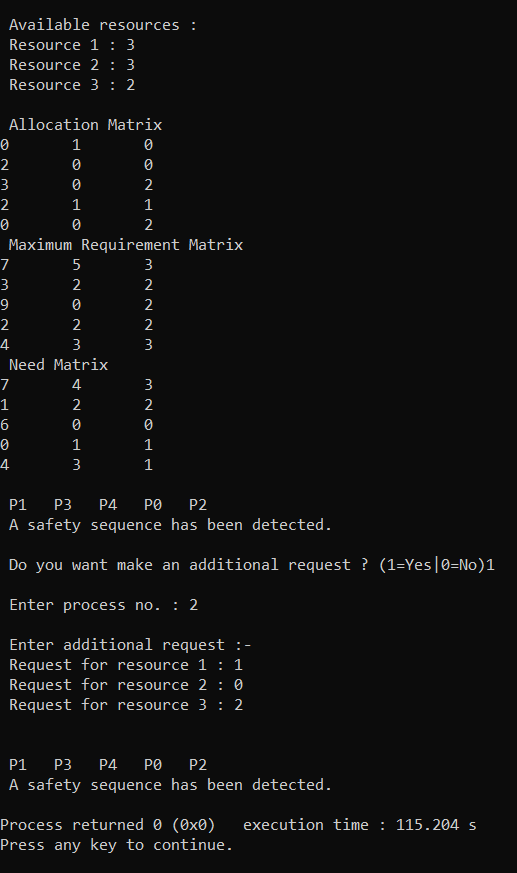
exit(0);

return 0;

}

**Output(screenshots):**





**Inference:** Hence Safety Sequence is detected and Addition Resource Request is granted.