**Deadlock Avoidance and Detection Scheme**

**1)Banker’s algorithm and resource request algorithm.**

///Banker's algorithm and resource request algorithm

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

typedef struct process

{

int act; ///Resources allocated

int max; ///Max REsources

} Pr;

void input(int np,int nr,Pr p1[np][nr],int \*available)

{

int i=0,j=0; ///No of processes ,No of resources

printf("Enter the number of resource avaialble: ");

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

{

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

if(i==nr-1) break;

}

printf("\nEnter the number of resources allocated to process\n");

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

{

printf("p%d: ",i);

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

{

scanf("%d",&p1[i][j].act); ///Enter number of resources type(j) allocated to i}

}

printf("\nEnter the number of maximum resources requested by process\n");

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

{

printf("p%d: ",i);

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

{

scanf("%d",&p1[i][j].max); ///Enter number of resources type(j) allocated to i}

}

}

void dd(int np,int nr,Pr p1[np][nr],int \*a,int need[np][nr])

{

int \*work=(int\*)malloc(sizeof(int)\*nr);

int\* ss=(int\*)malloc(sizeof(int)\*np);

int\* finish=(int\*)malloc(sizeof(int)\*np);

need[np][nr];

int i=0,j=0,sf=0; ///safe sequence index

printf("\n\_\_\_\_\_\_Need matrix is:\_\_\_\_\_\_\_ \n");

///Initialse the neeed matrix

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

{

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

{

need[i][j]=p1[i][j].max-p1[i][j].act;

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

}

printf("\n");

}

///Make work matrix equals avaiable matrix

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

{

work[i]=a[i];

}

///Make finish of all processes false(0)

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

{

finish[i]=0;

}

///initially assume there is deadlock so set dd=1 (true)

int dd=1;

do

{

int no\_i=1; ///No i is found yet

///Find if there exist nay process whose finish is false(0)

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

{

if(finish[i]==0)

{

///Check if request[i]<=work[i]

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

{

if(need[i][j]>work[j]) break;

}

///If j==nr means req[i]<=work[i] wrt all reasources

///Hence we set finish of this process to true

if(j==nr)

{

///work=work+allocated

ss[sf++]=i;

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

{

work[j]+=(p1[i][j].act);

}

finish[i]=1;

no\_i=0; ///We found an i with both conditions true

continue;

}

}

}

///If we had not found any i with both consition true

if(no\_i==1)

{

///and finish[i]==false(0) for any process means deadlock exists

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

{

if(finish[i]==0)

{

dd=1; ///deadlock exists => No safe sequence

break;

}

}

}

///Else if any processes were selected,means work was updated

else

{

int cont=0; ///Not relooping

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

{

///And we have some processes unfinihsd

///there are chances that they can finish now

if(finish[i]==0)

{

dd=0; ///No deadlock, as we are checking again

cont=1; ///Choosing to reloop

break;

}

}

///If cont==0 means all processes have executed withut any deadlock

if(cont==0)

{

dd=0; ///No dd => safe sequence exists

break;

}

}

}

while(!dd); ///Loop until no deadlock is found

if(dd)

{

printf("----------------------------------------------------");

printf("\nSystem is not in safe state hence no safe sequence");

printf("\n----------------------------------------------------\n");

}

else

{

printf("---------------------------------------------");

printf("\nSystem is in safe state , safe sequence is: ");

for(i=0; i<np; i++) printf(" p%d ",ss[i]);

printf("\n---------------------------------------------");

}

}

void res\_req(int np,int nr,Pr p[np][nr],int \*a,int need[np][nr])

{

int grant=0,c1=0,c2=0,i=0,pno; ///Request not yet granted, non of conditions satisfied yet

int\* req=(int\*)malloc(sizeof(int)\*nr);

printf("\nEnter the process no making the request: ");

scanf("%d",&pno);

printf("Enter the resources requested in order: ");

for(i=0; i<nr; i++) scanf("%d",&req[i]);

///check condition one i.e req(i)>=max(i)

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

{

///if process pno demands more resources of type i thn max

if(req[i]>p[pno][i].max) break;

}

///If condition is satisfied

if(i==nr) c1=1;

else

{

printf("----------------------------------------------------");

printf("\nProcess has exceeded it's maximum claim");

printf("\n----------------------------------------------------");

return;

}

///Check if we can grant the resources to process i.e req(i)<=avaiable(i)

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

{

///if process pno demands more resources of type i thn max

if(req[i]>a[i]) break;

}

///If consition is satisfied

if(i==nr) c2=1;

else

{

printf("------------------------------------------------------");

printf("\nResources are not available so request could not be granted");

printf("\n------------------------------------------------------");

return;

}

printf("----------------------------------------------------");

printf("\nRequest of process%d is granted!!",pno);

printf("\n----------------------------------------------------");

///Perform necessary updations

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

{

a[i]=a[i]-req[i];

need[pno][i]-=req[i];

p[pno][i].act+=req[i];

}

}

int main()

{

int np,nr;///No of processes ,No of resources

printf("Enter the number of processs: ");

scanf("%d",&np);

printf("Enter the number of resources: ");

scanf("%d",&nr);

Pr p1[np][nr];

int need[np][nr];

int\*avaiable=(int\*)malloc(sizeof(int)\*nr);

input(np,nr,p1,avaiable);;

int ch=1;

while(ch)

{

printf("\nChoose your choice: \n0 -> To exit\n1 -> To find safe sequence\n2 -> To make a resource request\nYour choice: ");

scanf("%d",&ch);

if(ch==0) break;

switch(ch)

{

case 1:

dd(np,nr,p1,avaiable,need);

break;

case 2:

res\_req(np,nr,p1,avaiable,need);

break;

default:

printf("\nInvalid input");

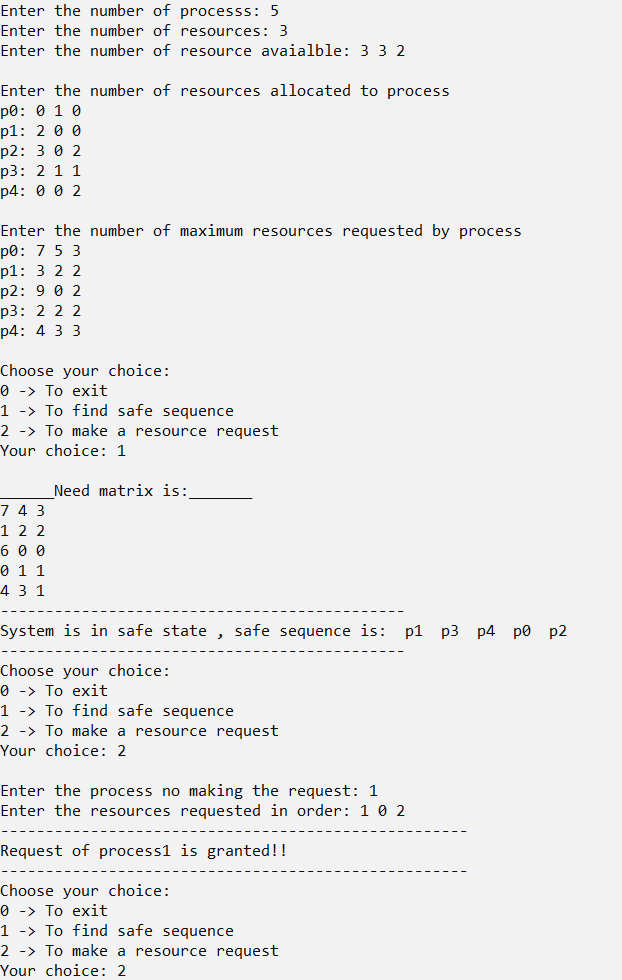
}

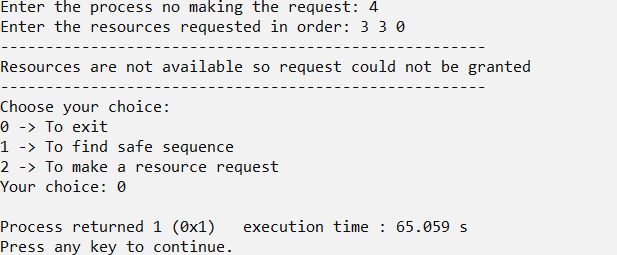
}

return 1;

}

**Output:**

****

****

**2)Deadlock Detection algorithm**

///Deadlock detection algorithm

///Adnan Ismail Shah Muzavor

#include<stdio.h>

#include<stdlib.h>

typedef struct process

{

int act; ///Resources allocated

int req; ///REsources requested

} Pr;

void input(int np,int nr,Pr p1[np][nr],int \*available)

{

int i=0,j=0; ///No of processes ,No of resources

printf("Enter the number of resource avaialble: ");

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

{

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

if(i==nr-1) break;

}

printf("\nEnter the number of resources allocated to process\n");

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

{ printf("p%d: ",i);

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

{

scanf("%d",&p1[i][j].act); ///Enter number of resources type(j) allocated to i}

}

printf("\nEnter the number of maximum resources requested by process\n");

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

{

printf("p%d: ",i);

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

{scanf("%d",&p1[i][j].req); ///Enter number of resources type(j) allocated to i}

}

}

}

void dd(int np,int nr,Pr p1[np][nr],int \*a)

{

int \*work=(int\*)malloc(sizeof(int)\*nr);

int\* finish=(int\*)malloc(sizeof(int)\*np);

int i=0,j=0;

///Make work matrix equals avaiable matrix

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

{work[i]=a[i];}

///Make finish of all processes false(0)

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

{finish[i]=0;}

///initially assume there is deadlock so set dd=1 (true)

int dd=1;

do

{ int no\_i=1; ///No i is found yet

///Find if there exist nay process whose finish is false(0)

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

{

if(finish[i]==0)

{

///Check if request[i]<=work[i]

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

{

if(p1[i][j].req>work[j])

{break;}

}

///If j==nr means req[i]<=work[i] wrt all reasources

///Hence we set finish of this process to true

if(j==nr)

{ ///work=work+allocated

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

{

work[j]+=(p1[i][j].act);

}

finish[i]=1;

no\_i=0; ///We found an i with both conditions true

continue;

}

}

}

///If we had not found any i with both consition true

if(no\_i==1)

{ ///and finish[i]==false(0) for any process means deadlock exists

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

{

if(finish[i]==0)

{ dd=1; ///deadlock exists

break;

}

}

}

///Else if any processes were selected,means work was updated

else

{

int cont=0; ///Not relooping

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

{

///And we have some processes unfinihsd

///there are chances that they can finish now

if(finish[i]==0)

{ dd=0; ///No deadlock, as we are checking again

cont=1; ///Choosing to reloop

break;

}

}

///If cont==0 means all processes have executed withut any deadlock

if(cont==0)

{ dd=0; ///No dd

break;

}

}

}

while(!dd); ///Loop until no deadlock is found

if(dd)

{ printf("----------------------------------------------------");

printf("\nThere is deadlock in the system.");

printf("\n----------------------------------------------------");

}

else

{ printf("----------------------------------------------------");

printf("\nThere is no deadlock in the system.");

printf("\n----------------------------------------------------");

}

}

int main()

{

int np,nr;///No of processes ,No of resources

printf("Enter the number of processs: ");

scanf("%d",&np);

printf("Enter the number of resources: ");

scanf("%d",&nr);

Pr p1[np][nr];

int\*available=(int\*)malloc(sizeof(int)\*nr);

input(np,nr,p1,available);

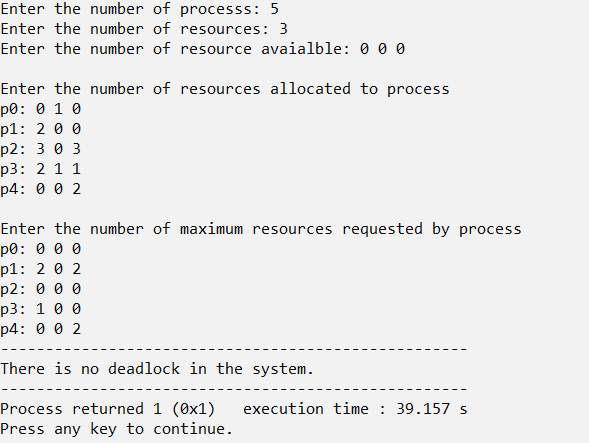
dd(np,nr,p1,available);

return 1;

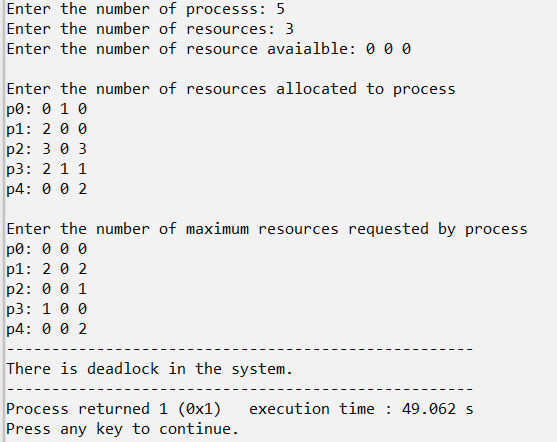
}

**Output:**

**No deadlock case:**

****

**Deadlock present case:**

****