Name : Sarthak Pagar

Roll No. : 34

Classs : TE (IT)

Practical : 5

Statement : [Implement the C program for Deadlock Avoidance Algorithm: Bankers Algorithm.](https://drive.google.com/file/d/1HyomGF17N_1KykfI10JWNV0c3uxdscub/view?usp=sharing)

#include<stdio.h>

struct process{

int max[10],allocate[10],need[10];

}p[10];

int n,m;

void input(int[]);

void display();

int isSafestate(int[],int[]);

int safetyalgorithm(int[],int[]);

int main(){

int i;

printf("\nEnter No of processes: ");

scanf("%d",&n);

printf("Enter no of resources: ");

scanf("%d",&m);

int available[m];

int safesequence[n];

printf("\n\*\*\*\*\*Enter details of process\*\*\*\*\*");

input(available);

display();

if(isSafestate(available,safesequence))

{

printf("\n\tSYSTEM IS IN SAFE STATE...");

printf("\nsafesequence is: ");

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

printf("P%d -> ",safesequence[i]);

}

else

printf("\nSYSTEM IS IN UNSAFE STATE!!!");

return 0;

}

void input (int available[m]){

int i,j;

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

printf("\nEnter the details of process P%d: ",i);

printf("\n\tEnter the allocated resources: ");

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

{

scanf("%d",&p[i].allocate[j]);

}

printf("\tEnter the max resources: ");

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

{

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

p[i].need[j]=p[i].max[j]-p[i].allocate[j];

}

}

printf("\nEnter the available resources: ");

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

{

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

}

}

void display()

{

int i,j;

printf("\n\tPID\tALLOCATE\tMAX\t\tNEED\n");

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

{

printf("\tP%d\t",i);

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

printf("%d ",p[i].allocate[j]);

printf("\t\t");

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

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

printf("\t\t");

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

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

printf("\n");

}

}

int isSafestate(int available[m],int safesequence[n])

{

if(safetyalgorithm(available,safesequence)==1)

return 1;

return 0;

}

int safetyalgorithm(int available[m],int safesequence[n])

{

int i,j;

int work[m],finish[n];

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

work[j]=available[j];

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

finish[i]=0;

int proceed=1,k=0;

while(proceed)

{

proceed=0;

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

{

int flag=1;

if(finish[i]==0)

{

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

{

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

{

continue;

}

else

{

flag=0;

break;

}

}

if(flag==0)

continue;

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

{

work[j]+=p[i].allocate[j];

}

finish[i]=1;

safesequence[k++]=i;

proceed=1;

}

}

}

for(i=0;i<n&&finish[i]==1;i++)

continue;

if(i==n)

return 1;

return 0;

}

Output :-

[Saru1594@localhost 5]$ gcc Banker.c

[Saru1594@localhost 5]$ ./a.out

Enter No of processes: 5

Enter no of resources: 3

\*\*\*\*\*Enter details of process\*\*\*\*\*

Enter the details of process P0:

Enter the allocated resources: 0 1 0

Enter the max resources: 7 5 3

Enter the details of process P1:

Enter the allocated resources: 2 0 0

Enter the max resources: 3 2 2

Enter the details of process P2:

Enter the allocated resources: 3 0 2

Enter the max resources: 9 0 2

Enter the details of process P3:

Enter the allocated resources: 2 1 1

Enter the max resources: 2 2 2

Enter the details of process P4:

Enter the allocated resources: 0 0 2

Enter the max resources: 4 3 3

Enter the available resources: 3 3 2

PID ALLOCATE MAX NEED

P0 0 1 0 7 5 3 7 4 3

P1 2 0 0 3 2 2 1 2 2

P2 3 0 2 9 0 2 6 0 0

P3 2 1 1 2 2 2 0 1 1

P4 0 0 2 4 3 3 4 3 1

SYSTEM IS IN SAFE STATE...

safesequence is: P1 -> P3 -> P4 -> P0 -> P2 ->