Consider the following snapshot of a system:

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Allocation | Max | Available |
|  | A B C D | A B C D | A B C D |
| P0 | 3 0 1 4 | 5 1 1 7 | 0 3 0 1 |
| P1 | 2 2 1 0 | 3 2 1 1 |  |
| P2 | 3 1 2 1 | 3 3 2 1 |  |
| P3 | 0 5 1 0 | 4 6 1 2 |  |
| P4 | 4 2 1 2 | 6 3 2 5 |  |

Write a C program that implements Banker’s Algorithm, to determine if the given snapshot of the system is in a safe state. Illustrate the order in which the process may complete, or illustrate why the state is unsafe.

#include<stdio.h>

int main()

{

int p,r;

int count=0;

int i,j;

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

int safe[10];

int avail[5];

int done[5];

int finish=0;

printf(“Enter the number of process and resources\n”);

scanf(“%d%d”,&p,&r);

printf(“Enter the allocation of resources of all process %dx%d matrix\n”,p,r);

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

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

scanf(“%d”,&aloc[i][j]);

printf(“Enter the max of resources of all process %dx%d matrix\n”,p,r);

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

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

scanf(“%d”,&max[i][j]);

printf(“Enter the available resources \n”);

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

scanf(“%d”,&avail[i]);

printf(“Enter the need matrix\n”);

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

{

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

{

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

printf(“%d\t”,need[i][j]);

}

printf(“\n”);

}

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

done[i]=0;

while(count<p)

{

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

{

if(done[i]==0)

{

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

{

if(need[i][j]>avail[j])

break;

}

if(j==r)

{

safe[count]=i;

done[i]=1;

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

{

avail[j]+=aloc[i][j];

}

count++;

finish=0;

}

else

{

finish++;

}

}

}

if(finish==(p-1))

{

printf(“Safe sequence does not exist”);

break;

}

}

if(finish!=(p-1))

{

printf(“\n available resource after completion\n”);

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

{

printf(“%d\t”,avail[i]);

}

printf(“\n Safe Sequence are\n”);

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

{

printf(“%d\t”,safe[i]);

}

}

return 0;

}