```
Enter details for P3
Enter allocation          --    2 1 1
Enter Max                 --    2 2 2
Enter details for P4
Enter allocation          --    0 0 2
Enter Max                 --    4 3 3

Enter Available Resources      --    3 3 2

Enter New Request Details --
Enter pid          --     1
Enter Request for Resources    --    1 0 2

P1 is visited(  5  3  2)
P3 is visited(  7  4  3)
P4 is visited(  7  4  5)
P0 is visited(  7  5  5)
P2 is visited( 10  5  7)
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2 )
Process         Allocation              Max                     Need
P0          0    1    0    7    5    3    7    4    3
P1          3    0    2    3    2    2    0    2    0
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

Process returned 5 (0x5)   execution time : 81.658 s
Press any key to continue.
```

#include<stdio.h>

struct file

{

int all[10];

int max[10];

int need[10];

int flag;

};

void main()

{

struct file f[10];

int fl;

int i, j, k, p, b, n, r, g, cnt=0, id, newr;

int avail[10],seq[10];

printf("Enter number of processes -- ");

```c
scanf("%d",&n);

printf("Enter number of resources -- ");

scanf("%d",&r);

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

{

printf("Enter details for P%d",i);

printf("\nEnter allocation\t -- \t");

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

scanf("%d",&f[i].all[j]);

printf("Enter Max\t\t -- \t");

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

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

f[i].flag=0;

}

printf("\nEnter Available Resources\t -- \t");

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

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

printf("\nEnter New Request Details -- ");

printf("\nEnter pid \t -- \t");

scanf("%d",&id);

printf("Enter Request for Resources \t -- \t");

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

{

scanf("%d",&newr);

f[id].all[i] += newr;

avail[i]=avail[i] - newr;

}

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

{

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

{
```

```c
f[i].need[j]=f[i].max[j]-f[i].all[j];

if(f[i].need[j]<0)

f[i].need[j]=0;

}

}

cnt=0;

fl=0;

while(cnt!=n)

{

g=0;

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

{

if(f[j].flag==0)

{

b=0;

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

{

if(avail[p]>=f[j].need[p])

b=b+1;

else

b=b-1;

}

if(b==r)

{

printf("\nP%d is visited",j);

seq[fl++]=j;

f[j].flag=1;

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

avail[k]=avail[k]+f[j].all[k];

cnt=cnt+1;

printf("(");
```

```c
for(k=0;k<r;k++)
printf("%3d",avail[k]);
printf(")");
g=1;
}
}
}
if(g==0)
{
printf("\n REQUEST NOT GRANTED -- DEADLOCK OCCURRED");
printf("\n SYSTEM IS IN UNSAFE STATE");
goto y;
}
}
printf("\nSYSTEM IS IN SAFE STATE");
printf("\nThe Safe Sequence is -- (");
for(i=0;i<fl;i++)
printf("P%d ",seq[i]);
printf(")");
y: printf("\nProcess\t\tAllocation\t\tMax\t\t\tNeed\n");
for(i=0;i<n;i++)
{
printf("P%d\t",i);
for(j=0;j<r;j++)
printf("%6d",f[i].all[j]);
for(j=0;j<r;j++)
printf("%6d",f[i].max[j]);
for(j=0;j<r;j++)
printf("%6d",f[i].need[j]);
printf("\n");
}
```

}

```
Enter details for P3
Enter allocation         --     2 1 1
Enter Max                --     2 2 2
Enter details for P4
Enter allocation         --     0 0 2
Enter Max                --     4 3 3

Enter Available Resources        --     3 3 2

Enter New Request Details --
Enter pid          --     0
Enter Request for Resources      --     0 0 0

P1 is visited(  5  3  2)
P3 is visited(  7  4  3)
P4 is visited(  7  4  5)
P0 is visited(  7  5  5)
P2 is visited( 10  5  7)
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2 )
Process        Allocation            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

Process returned 5 (0x5)   execution time : 107.591 s
Press any key to continue.
```