5.Illustrate the deadlock avoidance concept by simulating Banker’s algorithm with C.

#include<stdio.h>

void main()

{

int n,r,i,j,k,p,u=0,s=0,m;

int block[10],run[10],active[10],newreq[10];

int max[10][10],resalloc[10][10],resreq[10][10];

int totalloc[10],totext[10],simalloc[10];

//clrscr();

printf("Enter the no of processes:");

scanf("%d",&n);

printf("Enter the no ofresource classes:");

scanf("%d",&r);

printf("Enter the total existed resource in each class:");

for(k=1; k<=r; k++)

scanf("%d",&totext[k]);

printf("Enter the allocated resources:");

for(i=1; i<=n; i++)

for(k=1; k<=r; k++)

scanf("%d",&resalloc);

printf("Enter the process making the new request:");

scanf("%d",&p);

printf("Enter the requested resource:");

for(k=1; k<=r; k++)

scanf("%d",&newreq[k]);

printf("Enter the process which are n blocked or running:");

for(i=1; i<=n; i++)

{

if(i!=p)

{

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

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

}

}

block[p]=0;

run[p]=0;

for(k=1; k<=r; k++)

{

j=0;

for(i=1; i<=n; i++)

{

totalloc[k]=j+resalloc[i][k];

j=totalloc[k];

}

}

for(i=1; i<=n; i++)

{

if(block[i]==1||run[i]==1)

active[i]=1;

else

active[i]=0;

}

for(k=1; k<=r; k++)

{

resalloc[p][k]+=newreq[k];

totalloc[k]+=newreq[k];

}

for(k=1; k<=r; k++)

{

if(totext[k]-totalloc[k]<0)

{

u=1;

break;

}

}

if(u==0)

{

for(k=1; k<=r; k++)

simalloc[k]=totalloc[k];

for(s=1; s<=n; s++)

for(i=1; i<=n; i++)

{

if(active[i]==1)

{

j=0;

for(k=1; k<=r; k++)

{

if((totext[k]-simalloc[k])<(max[i][k]-resalloc[i][k]))

{

j=1;

break;

}

}

}

if(j==0)

{

active[i]=0;

for(k=1; k<=r; k++)

simalloc[k]=resalloc[i][k];

}

}

m=0;

for(k=1; k<=r; k++)

resreq[p][k]=newreq[k];

printf("Deadlock willn't occur");

}

else

{

for(k=1; k<=r; k++)

{

resalloc[p][k]=newreq[k];

totalloc[k]=newreq[k];

}

printf("Deadlock will occur");

}

}