```c
#include<stdio.h>
#include<stdlib.h>

#define MAX 10

void input(int,int);
void display();
int safestate(int,int);
void checkrequest();


int allocate[MAX][MAX];
int maximum[MAX][MAX];
int need[MAX][MAX];


int available[MAX];
int work[MAX];
int finish[MAX];
int sequence[MAX];
int request[MAX];

int i,j,n,r;

void main()
{
 int ch,ans;
 clrscr();
     do{
    printf("\n1 Accept Data \n2 Display Data \n3 Safety algorithm \n4 Request
    algorithm \n5 Exit\n");
    printf("\nEnter choice:");
    scanf("%d",&ch);
    switch(ch)
          {
        case 1: printf("\nEnter the no. of processes: ");
            scanf("%d",&n);
            printf("Enter the no. of resources: ");
            scanf("%d",&r);

            if(r>n)
              printf("\nResourrses can't be greater than no of processes");
            else
              input(n,r);
            break;

        case 2: display();
            break;

        case 3: ans=safestate(n,r);
            if(ans==1)
              {
               printf("\nSystem is in a safe state\n");
               printf("Safe sequence is: \n");
               for(i=0;i<n;i++)
                   printf("P%d ",sequence[i]);
              }
            else
               printf("\nSystem is not in a safe state\n");
            break;
        case 4: checkrequest();
            break;
        case 5: exit(0);
            break;
          }
     }while(ch!=5);
}

void input(int n,int r)
{
 printf("\nEnter initial allocation:");
 for(i=0;i<n;i++)
    {
     printf("\nEnter %d allocations for P%d:",r,i);
     for(j=0;j<r;j++)
     scanf("%d",&allocate[i][j]);
    }

 printf("\nEnter max Requirement:");
 for(i=0;i<n;i++)
    {
     printf("\nEnter %d max requirement for P%d:",r,i);
     for(j=0;j<r;j++)
    scanf("%d",&maximum[i][j]);
    }

 printf("Enter available Resources \n");
 for(i=0;i<r;i++)
    scanf("%d",&available[i]);


 for(i=0;i<n;i++)
    for(j=0;j<r;j++)
    need[i][j]=maximum[i][j]-allocate[i][j];

}

void display()
{
 printf("\nAllocation Matrix \n");
 for(i=0;i<n;i++)
    {
     for(j=0;j<r;j++)
     printf("\t%d ",allocate[i][j]);
     printf("\n");
    }
 printf("\nMax Matrix \n");
 for(i=0;i<n;i++)
    {
     for(j=0;j<r;j++)
     printf("\t%d ",maximum[i][j]);
     printf("\n");
    }

 printf("\nNeed Matrix \n");
 for(i=0;i<n;i++)
    {
     for(j=0;j<r;j++)
     printf("\t%d ",need[i][j]);
     printf("\n");
    }

 printf("Available Resources are [ ");
 for(i=0;i<r;i++)
    printf("%d ",available[i]);
 printf("]");

}

int safestate(int n,int r)
{
 int index=0,flag=1,cnt;

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

 for(i=0; i<r; i++)
     work[i]=available[i];


 //need must not be negative
 for(i=0;i<n;i++)
    for(j=0;j<r;j++)
    {
     if(need[i][j]<0)
       {
        printf("\n Allocated resources exceed maximum needs of P%d",i);
        return 0;
       }
    }

 while(flag)
      {
    flag=0;
    for(i=0;i<n;i++)
       {
        if(finish[i]==0)
          {
           cnt=0;
           for(j=0;j<r;j++)
          {
          if(need[i][j]<=work[j])
             cnt++;
          else
             break;
          }
           if(cnt==r)
         {
          finish[i]=1;
          for(j=0;j<r;j++)
             work[j]+=allocate[i][j];

          flag=1;
          sequence[index]=i;
          index++;
          }
           }
       }
      }

 for(i=0;i<n;i++)
    if(finish[i]==0)
       return 0;

 return 1;
}

void checkrequest()
{
 int c1=1,c2=1,p;

 printf("\nAvailable resources are: [ ");
 for(i=0;i<r;i++)
     printf("%d ", available[i]);
 printf("]");

 printf("\nEnter the requesting process: P");
 scanf("%d",&p);

 printf("Enter requests for P%d: ", p);
 for(i=0;i<r;i++)
     scanf("%d",&request[i]);

 for(i=0;i<r;i++)
    if(request[i]>available[i])
      {
       c1=0;
       printf("\nRequest by P%d exceeds the available resources\nIt cannot be
       immediately granted\n",p);
       break;
      }
 for(i=0;i<r;i++)
    if(request[i]>need[p][i])
      {
       c2=0;
       printf("\nRequest by P%d exceeds its maximum needs\nInvalid
       request\n",p);
       break;
      }

 if(c1 && c2)
   {
    for(i=0;i<r;i++)
       {
    available[i]-=request[i];
    allocate[p][i]+=request[i];
    need[p][i]-=request[i];
       }

    if(!safestate(n,r))
      {
       printf("\nSystem is in an unsafe state.\n");
       printf("Request by P%d cannot be immediately granted\n", p);
       for(i=0;i<r;i++)
      {
       available[i]+=request[i];
       allocate[p][i]-=request[i];
       need[p][i]+=request[i];
      }
       }
    else
       {
    printf("\nSystem is in a safe state");
    printf("\nRequest by P%d can be immediately granted\n",p);
       }

   }
   else
       printf("\nSystem is in an unsafe state\n");
}
```