```c
#include<stdio.h>

struct process
{
 int all[6],max[6],need[6],finished,request[6];
}p[10];
int avail[6],sseq[10],ss=0,check1=0,check2=0,n,pid,work[6];
int nor;
int safeseq(void);

void main()
{
int ch,i=0,j=0,k,pid,ch1;
int violationcheck=0,waitcheck=0;
do
{

 printf("\n\n\t 1. Input");
                printf("\n\n\t 2. New Request");
                printf("\n\n\t 3. Safe State or Not");
                printf("\n\n\t 4. print");
                printf("\n\n\t 5. Exit");
                printf("\n\n\t Enter ur choice :");
                scanf("%d",&ch);
 switch(ch)
 {
  case 1: printf("\n\n\t Enter number of processes : ");
        scanf("%d",&n);
        printf("\n\n\t Enter the Number of Resources : ");
        scanf("%d",&nor);
        printf("\n\n\t Enter the Available Resouces : ");
        for(j=0;j<n;j++)
        {
          for(k=0;k<nor;k++)
          {
           if(j==0)
            {
             printf("\n\n\t For Resource type %d : ",k);
             scanf("%d",&avail[k]);
            }
           p[j].max[k]=0;
           p[j].all[k]=0;
           p[j].need[k]=0;
           p[j].finished=0;
           p[j].request[k]=0;
          }
       }
      for(i=0;i<n;i++)
      {
        printf("\n\n\t Enter Max and Allocated resources for P%d : ",i);
        for(j=0;j<nor;j++)
        {
           printf("\n\n\t Enter the Max of resource %d : ",j);
           scanf("%d",&p[i].max[j]);
           printf("\n\n\t Allocation of resource %d    :",j);
           scanf("%d",&p[i].all[j]);
           if(p[i].all[j]>p[i].max[j])
           {
                printf("\n\n\t Allocation should be less < or =max");
                j--;
           }
           else
           {
                p[i].need[j]=p[i].max[j]-p[i].all[j];
                avail[j]=avail[j]-p[i].all[j];
           }
```

```
        }
 }
break;
case 2: violationcheck=0;
        waitcheck=0;
        printf("\n\n\t Requesting process id :");
        scanf("%d",&pid);
        for(j=0;j<nor;j++)
        {
          printf("\n\n\t Number of Request for resource %d :",j);
          scanf("%d",&p[pid].request[j]);
          if(p[pid].request[j]>p[pid].need[j])
          violationcheck=1;
          if(p[pid].request[j]>avail[j])
          waitcheck=1;
        }

        if(violationcheck==1)
        printf("\n\n\t The Process Exceeds its Max Need: Terminated");
        else if(waitcheck==1)
        printf("\n\n\t Lack of Resourcess : cannot be granted");
        else
        {
                for(j=0;j<nor;j++)
                {
                        avail[j]=avail[j]-p[pid].request[j];
                        p[pid].all[j]=p[pid].all[j]+p[pid].request[j];
                        p[pid].need[j]=p[pid].need[j]-p[pid].request[j];
                }
                ch1=safeseq();
                if(ch1==0)
                {
                        for(j=0;j<nor;j++)
                        {
                        avail[j]=avail[j]+p[pid].request[j];
                        p[pid].all[j]=p[pid].all[j]-p[pid].request[j];
                        p[pid].need[j]=p[pid].need[j]+p[pid].request[j];
                        }
                }
                else if(ch1==1)
                printf("\n\n\t Request can be granted ");
        }
        break;
case 3:  if(safeseq()==1)
          printf("\n\n\t The System is in safe state ");
         else
         printf("\n\n\t The System is Not in safe state ");
         break;
case 4: printf("\n\n\t Number of processes : %d",n);
        printf("\n\n\t Number of Resources : %d",nor);
        printf("\n\n\t Pid\tMax\tAllocated\tNeed ");
        for(i=0;i<n;i++)
          {
                printf("\n\n\t  P%d : ",i);
                for(j=0;j<nor;j++)
                        printf("%d",p[i].max[j]);
                printf("\t");
                for(j=0;j<nor;j++)
                        printf("%d",p[i].all[j]);
                printf("\t\t");
                for(j=0;j<nor;j++)
                        printf("%d",p[i].need[j]);
          }
           printf("\n\n\t Available :");
          for(i=0;i<nor;i++)
             printf("%d",avail[i]);
```

```
          break;
case 5: break;
}   //End of switch
}while(ch!=5);
}   //End of main

int safeseq()
{
        int tj,tk,i,j,k;
        ss=0;
         for(j=0;j<nor;j++)
                work[j]=avail[j];
        for(j=0;j<n;j++)
                p[j].finished=0;
        for(tk=0;tk<nor;tk++)
        {
                for(j=0;j<n;j++)
                {
                         if(p[j].finished==0)
                        {
                                check1=0;
                                for(k=0;k<nor;k++)
                                        if(p[j].need[k]<=work[k])
                                                check1++;
                                if(check1==nor)
                                {
                                        for(k=0;k<nor;k++)
                                        {
                                                work[k]=work[k]+p[j].all[k];
                                                 p[j].finished=1;
                                        }
                                         sseq[ss]=j;
                                        ss++;
                                }
                        }
                } //End of j loop
         } //End of tk loop
 check2=0;
        for(i=0;i<n;i++)
         if(p[i].finished==1)
                check2++;
        printf("\n\n\t");
        if(check2>=n)
        {
                printf("\n\n\t The safe sequence is:\t");
                for( tj=0;tj<n;tj++)
                        printf("P%d,",sseq[tj]);
                return 1;
        }
        else
                return 0;
  }
```