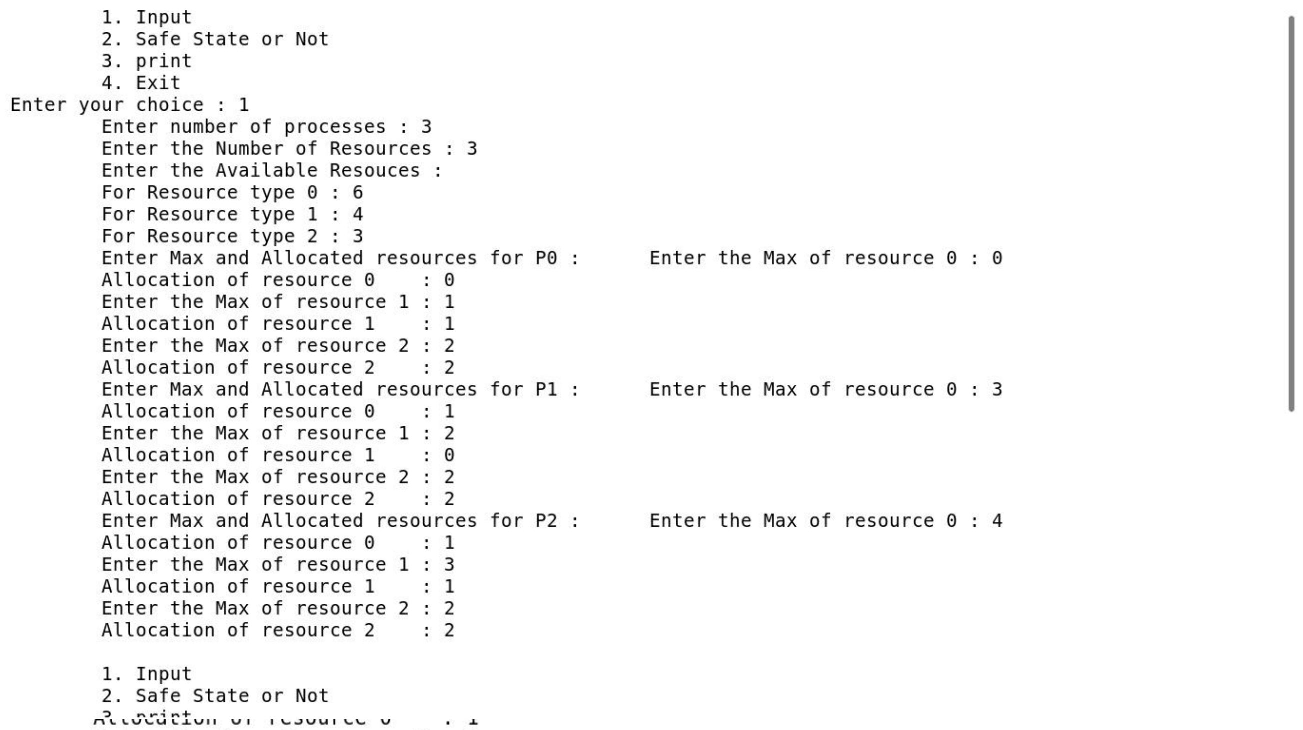
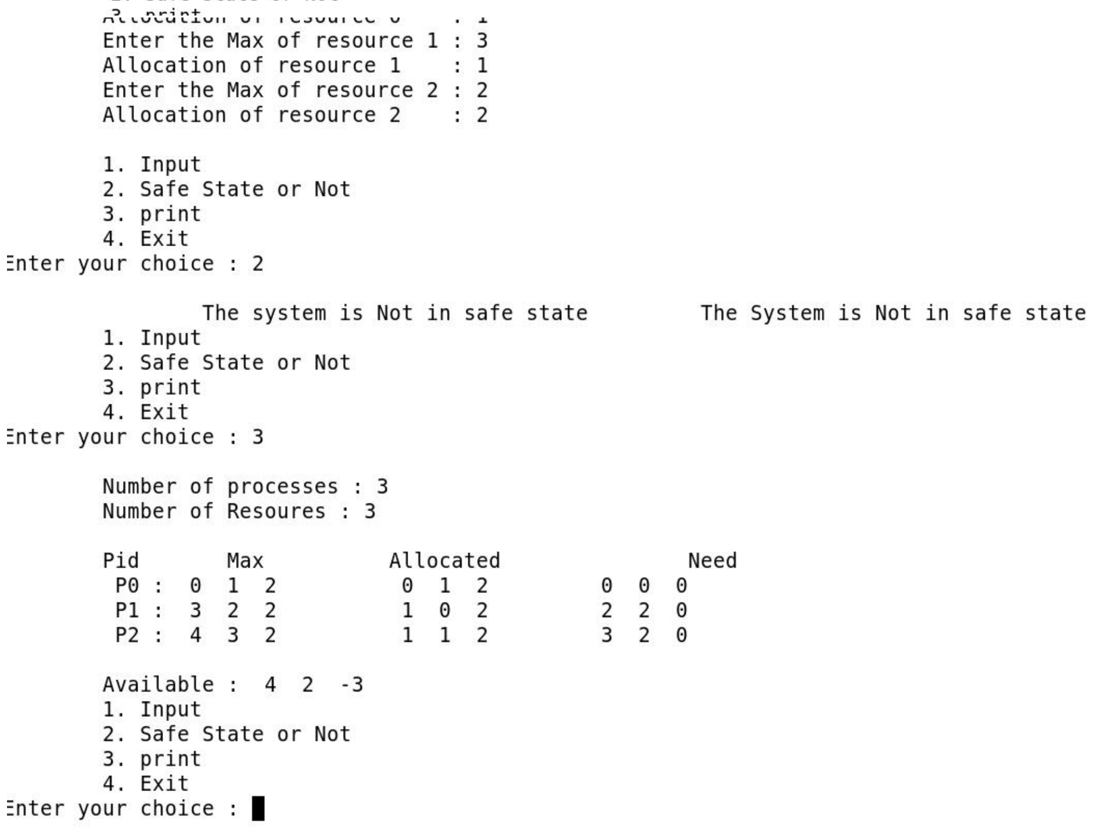
```c
#include <stdio.h>
struct process
{
int all[6], max[6], need[6], finished, request[6];
} p[10];
int avail[6], sseq[10], ss = 0, check1 = 0, check2 = 0, n, pid,
work[6];
int nor, nori;
int main()
{
int safeseq(void);
int ch, i = 0, j = 0, k, pid, ch1;
int violationcheck = 0, waitcheck = 0;
do
{
// clrscr();
printf("\n\t 1. Input");
printf("\n\t 2. Safe State or Not");
printf("\n\t 3. print");
printf("\n\t 4. Exit");
printf("\n Enter your choice : ");
scanf("%d", &ch);
switch (ch){
case 1:
printf("\t Enter number of processes : ");
scanf("%d", &n);
printf("\t Enter the Number of Resources : ");
scanf("%d", &nor);
printf("\t Enter the Available Resouces : \n");
for (j = 0; j < n; j++)
{for (k = 0; k < nor; k++){
if (j == 0){
printf("\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("\t Enter Max and Allocated resources for P%d : ", i);
for (j = 0; j < nor; j++){
printf("\t Enter the Max of resource %d : ", j);
scanf("%d", &p[i].max[j]);
printf("\t Allocation of resource %d : ", j);
scanf("%d", &p[i].all[j]);
if (p[i].all[j] > p[i].max[j])
{
printf("\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:
if (safeseq() == 1)
printf("\t The System is in safe state ");
else
printf("\t The System is Not in safe state ");
break;
case 3:
printf("\n\t Number of processes : %d", n);
printf("\n\t Number of Resoures : %d", nor);
printf("\n\n\t Pid \t Max \t\tAllocated \t\tNeed ");
for (i = 0; i < n; i++){
printf("\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");
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 4:
return 0;
}
// getch();
} while (ch != 4);}
int safeseq()
{
int 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 (k = 0; k < nor; k++)
{
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++;
}}}}
check2 = 0;
for (i = 0; i < n; i++)
if (p[i].finished == 1)
check2++;
printf("\n\t");
if (check2 >= n)
{
printf("\t The system is in safe state");
for (j = 0; j < n; j++)
printf("P%d, ", sseq[j]);
return 1;
}
else
printf("\t The system is Not in safe state");
return 0;
}
```



```
        1. Input
        2. Safe State or Not
        3. print
        4. Exit
 Enter your choice : 1
        Enter number of processes : 3
        Enter the Number of Resources : 3
        Enter the Available Resouces :
        For Resource type 0 : 6
        For Resource type 1 : 4
        For Resource type 2 : 3
        Enter Max and Allocated resources for P0 :      Enter the Max of resource 0 : 0
        Allocation of resource 0    : 0
        Enter the Max of resource 1 : 1
        Allocation of resource 1    : 1
        Enter the Max of resource 2 : 2
        Allocation of resource 2    : 2
        Enter Max and Allocated resources for P1 :      Enter the Max of resource 0 : 3
        Allocation of resource 0    : 1
        Enter the Max of resource 1 : 2
        Allocation of resource 1    : 0
        Enter the Max of resource 2 : 2
        Allocation of resource 2    : 2
        Enter Max and Allocated resources for P2 :      Enter the Max of resource 0 : 4
        Allocation of resource 0    : 1
        Enter the Max of resource 1 : 3
        Allocation of resource 1    : 1
        Enter the Max of resource 2 : 2
        Allocation of resource 2    : 2

        1. Input
        2. Safe State or Not
        3. print
```



```
        Allocation of resource 0    : 1
        Enter the Max of resource 1 : 3
        Allocation of resource 1    : 1
        Enter the Max of resource 2 : 2
        Allocation of resource 2    : 2

        1. Input
        2. Safe State or Not
        3. print
        4. Exit
nter your choice : 2

                The system is Not in safe state         The System is Not in safe state
        1. Input
        2. Safe State or Not
        3. print
        4. Exit
nter your choice : 3

        Number of processes : 3
        Number of Resoures : 3

        Pid       Max          Allocated              Need
         P0 :  0  1  2          0  1  2         0  0  0
         P1 :  3  2  2          1  0  2         2  2  0
         P2 :  4  3  2          1  1  2         3  2  0

        Available :  4  2  -3
        1. Input
        2. Safe State or Not
        3. print
        4. Exit
nter your choice : █
```