OS: BANKER'S ALGORITHM

VIBHA HUGAR CS255

WEEK 6 OS LAB

CODE

```
#include <stdio.h>
#include <conio.h>

int main()
{
int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];
int p, r, i, j, process, count;
count = 0;

printf("Enter the no of processes : ");
scanf("%d", &p);

for(i = 0; i< p; i++)
        completed[i] = 0;

printf("\n\nEnter the no of resources : ");
scanf("%d", &r);

printf("\n\nEnter the Max Matrix for each process : ");
for(i = 0; i < p; i++)
{
        printf("\nFor process %d : ", i + 1);
        for(j = 0; j < r; j++)
```

```
                scanf("%d", &Max[i][j]);
}

printf("\n\nEnter the allocation for each process : ");
for(i = 0; i < p; i++)
{
        printf("\nFor process %d : ",i + 1);
        for(j = 0; j < r; j++)
                scanf("%d", &alloc[i][j]);
}

printf("\n\nEnter the Available Resources : ");
for(i = 0; i < r; i++)
                scanf("%d", &avail[i]);



        for(i = 0; i < p; i++)
                for(j = 0; j < r; j++)
                        need[i][j] = Max[i][j] - alloc[i][j];

do
{
        printf("\n Max matrix:\tAllocation matrix:\n");
        for(i = 0; i < p; i++)
        {
                for( j = 0; j < r; j++)
                        printf("%d  ", Max[i][j]);
                printf("\t\t");
                for( j = 0; j < r; j++)
```

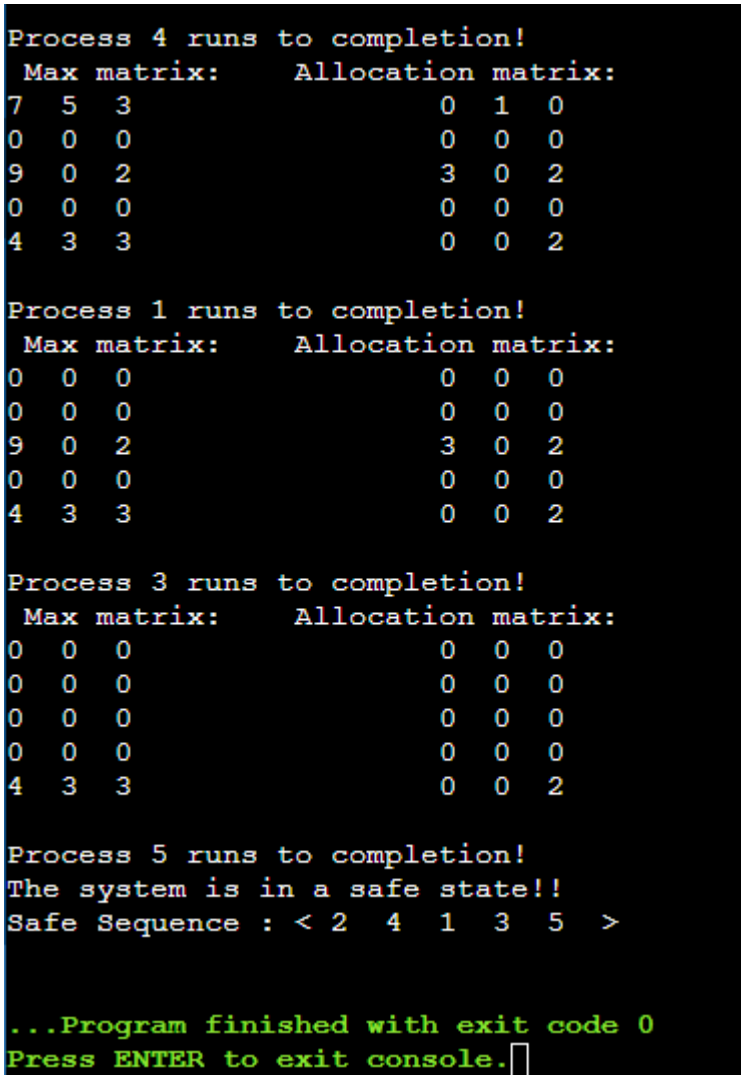
```
                printf("%d  ", alloc[i][j]);
        printf("\n");
}

process = -1;

for(i = 0; i < p; i++)
{
        if(completed[i] == 0)//if not completed
        {
                process = i ;
                for(j = 0; j < r; j++)
                {
                        if(avail[j] < need[i][j])
                        {
                                process = -1;
                                break;
                        }
                }
        }
        if(process != -1)
                break;
}

if(process != -1)
{
        printf("\nProcess %d runs to completion!", process + 1);
        safeSequence[count] = process + 1;
        count++;
```

```
                for(j = 0; j < r; j++)
                {
                        avail[j] += alloc[process][j];
                        alloc[process][j] = 0;
                        Max[process][j] = 0;
                        completed[process] = 1;
                }
        }
}while(count != p && process != -1);

if(count == p)
{
        printf("\nThe system is in a safe state!!\n");
        printf("Safe Sequence : < ");
        for( i = 0; i < p; i++)
                        printf("%d  ", safeSequence[i]);
        printf(">\n");
}
else
        printf("\nThe system is in an unsafe state!!");
getch();
}
```

OUTPUT

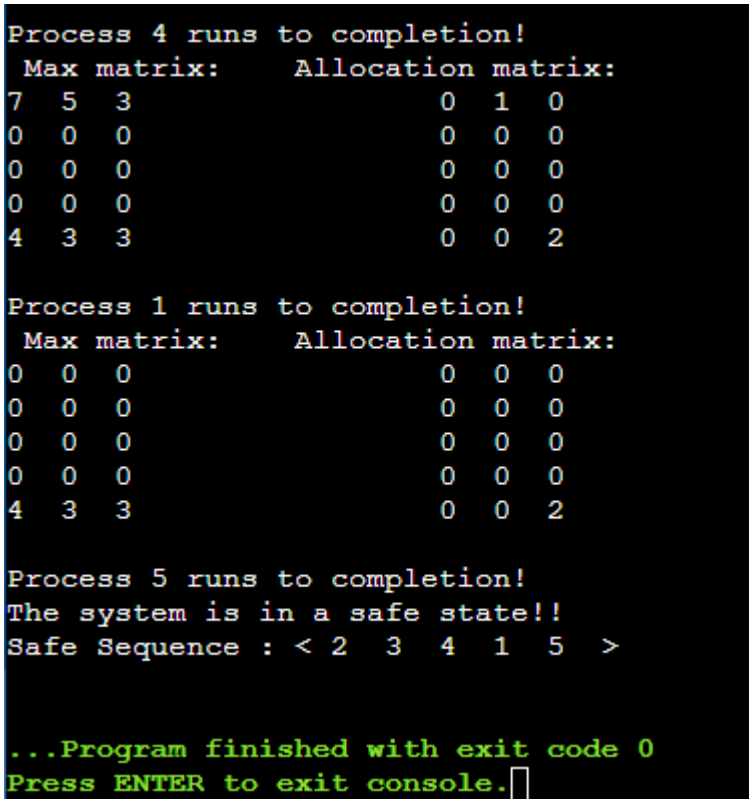
For the original given problem:

```
Enter the no of processes : 5


Enter the no of resources : 3


Enter the Max Matrix for each process :
For process 1 : 7
5
3

For process 2 : 3 2 2

For process 3 : 9 0 2

For process 4 : 2 2 2

For process 5 : 4 3 3


Enter the allocation for each process :
For process 1 : 0 1 0

For process 2 : 2 0 0

For process 3 : 3 0 2

For process 4 : 2 1 1

For process 5 : 0 0 2


Enter the Available Resources : 3 3 2

 Max matrix:    Allocation matrix:
7  5  3                0  1  0
3  2  2                2  0  0
9  0  2                3  0  2
2  2  2                2  1  1
4  3  3                0  0  2

Process 2 runs to completion!
 Max matrix:    Allocation matrix:
7  5  3                0  1  0
0  0  0                0  0  0
9  0  2                3  0  2
2  2  2                2  1  1
4  3  3                0  0  2
```



If P1 (2nd process) requests one additional instance of A and two of C:

```
Enter the no of processes : 5


Enter the no of resources : 3


Enter the Max Matrix for each process :
For process 1 : 7 5 3

For process 2 : 3 2 2

For process 3 : 9 0 2

For process 4 : 2 2 2

For process 5 : 4 3 3


Enter the allocation for each process :
For process 1 : 0 1 0

For process 2 : 3 0 2

For process 3 : 3 0 2

For process 4 : 2 1 1

For process 5 : 0 0 2


Enter the Available Resources : 3 3 2

 Max matrix:    Allocation matrix:
7  5  3                0  1  0
3  2  2                3  0  2
9  0  2                3  0  2
2  2  2                2  1  1
4  3  3                0  0  2

Process 2 runs to completion!
 Max matrix:    Allocation matrix:
7  5  3                0  1  0
0  0  0                0  0  0
9  0  2                3  0  2
2  2  2                2  1  1
4  3  3                0  0  2

Process 3 runs to completion!
 Max matrix:    Allocation matrix:
7  5  3                0  1  0
0  0  0                0  0  0
0  0  0                0  0  0
2  2  2                2  1  1
4  3  3                0  0  2
```



With work and finish vectors

CODE

```
printf("Is there any extra requirement from any of the Processes? -1 is no:\n");
  scanf("%d", &ch);

  if(ch != -1)
  {
     for(i = 0; i<n; i++){

        if(i == ch)
        {
           for(j = 0; j<m; j++)
           {
              printf("Enter %d Value of P%d", j+1, i);
```

```
            scanf("%d", &ch);
            alloc[i][j] += ch;
            avail[j]-=ch;
          }
        }
      }
    }
```

OUTPUT

(same output as initial output, we just need to enter the value as below)

```
Is there any extra requirement from any of the Processes? -1 is no:
-1
```

```
The system is in a safe state!!
Safe Sequence : < 2  4  1  3  5  >
```