Task 8

Task 8 Simulate Bankers Algorithm for Deadlock Avoidance in C.

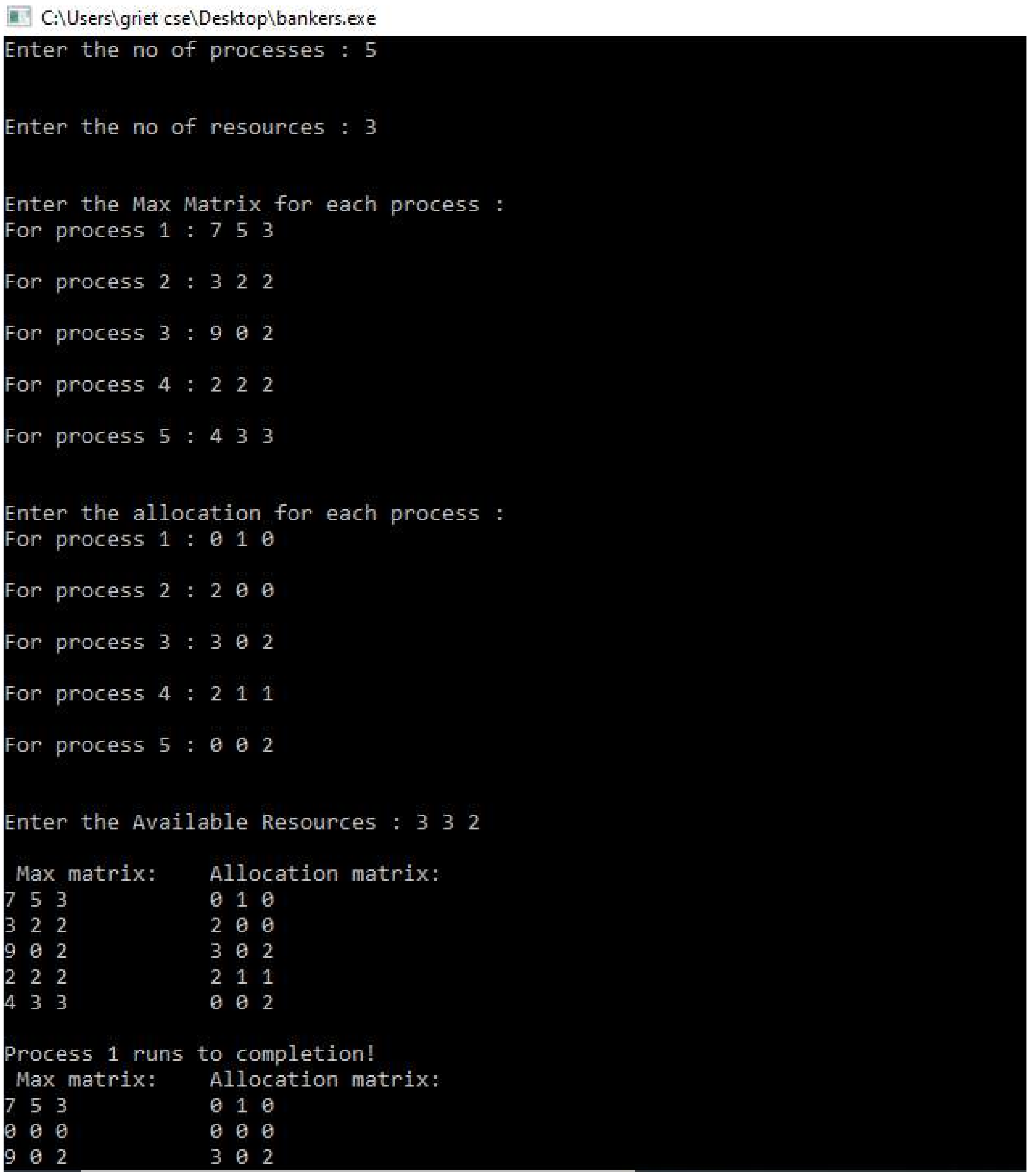
Program:

```
#include  <stdio.h>
#include <stdlib.h>
int main()
{
int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];
/*Max denotes max required resource
alloc denotes already allocated resouces for each process
avail denotes available resource of each kind
completed array indicates whether each process has met with its requirements and completed
or not.
Safe sequence is an array which holds order of execution that can result in completion of all
process*/
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; /*initially no process is completed*/
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]; // process still need these many resorces.
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; //indicates process can not completed.
          for(i = 0; i < p; i++)
          {
          if(completed[i] == 0)//if not completed.
          {
          process = i ; //ith process not yet completed.
          for(j = 0; j < r; j++)
             {
          if(avail[j] < need[i][j])
                {
          process = -1; //excess required which is not possible
          break;
                }
             }
          }/*end if*/
if(process != -1)
break; /* that means there exists a process that can complete its requirement*/
          }/*for end*/
/* process holds i th process which is not yet completed*/
if(process != -1)
{
printf("\nProcess %d runs to completion!", process );
safeSequence[count] = process ; /*join it to safe sequence*/
count++;    //identifying number of completed processes
for(j = 0; j < r; j++)
{
avail[j] += alloc[process][j]; /*return back the resources*/
alloc[process][j] = 0;
Max[process][j] = 0;
completed[process] = 1;
}
}
}while(count != p && process != -1); /*for all process*/

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
```

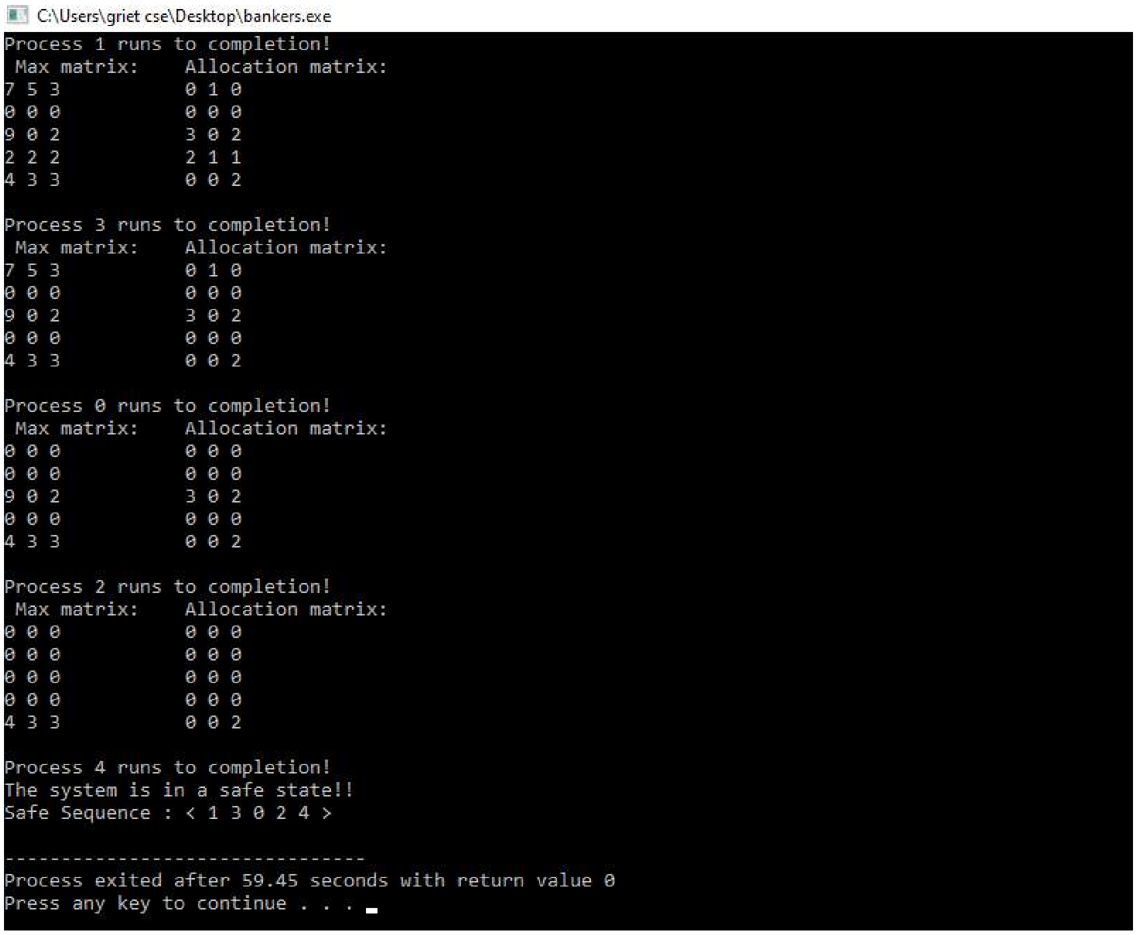
```
printf("\nThe system is in an unsafe state!!");
}
```

Output:



C:\Users\griet cse\Desktop\bankers.exe

```
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 1 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
```


C:\Users\griet cse\Desktop\bankers.exe

```
Process 1 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
9 0 2           3 0 2
0 0 0           0 0 0
4 3 3           0 0 2

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

Process 2 runs to completion!
 Max matrix:    Allocation matrix:
0 0 0           0 0 0
0 0 0           0 0 0
0 0 0           0 0 0
0 0 0           0 0 0
4 3 3           0 0 2

Process 4 runs to completion!
The system is in a safe state!!
Safe Sequence : < 1 3 0 2 4 >

--------------------------------
Process exited after 59.45 seconds with return value 0
Press any key to continue . . .
```