------------ 15CS01011_OSLAB6_RESONING ----------------

Part 2. BANKER’S ALGORITHM WITH DEADLOCK PREVENTION :

-> When there is a random request, the system determines whether allocating resources for the request leaves the system in a safe state that avoids deadlock. If no, then wait for another process to release resources.

-> each Process declares its maximum demands must be less than total resources in system.

-> Random requests are generated by the 4 process(threads).

-> A pthread mutex lock is used for the mutual exclusion of the 4 processes and to maitain the synchronization.

-> **4 semaphores** are used which are initialized to the system_resource_matrix(i.e the initial available matrix with the system). These semaphores are being waited when the resources are allocated to the process and are being released whenever the resources are released by that process.

-> **customer(void* Pi):** This function for process Pi contains a infinite loop where the requesting and releasing of Resources happen. Contains requestResouce() and releaseResource() functions.

-> **requestResource(Pi, requestVector):** This function checks if the request vector generated is in the need of that process and is less than or equal to the available resources. If every check is satisfied then it runs the safety algorithm for a safe sequence(i.e if this request is granted then is there a safe sequence). If yes, then the resources are allocated to that particular process. Else, whatever has been updated for safety check is rolled back.

-> **releaseResource(Pi, releaseVector, 0):** This function releases the resources once the need of all resources in a process becomes 0. And orderMat[] keeps the record of the processes that are completed.

-> **test_safety():** This function checks if there is a safe sequence for the

updated need matrix, available matrix. This returns 0 is safe else returns -1. If not safe, then the values changed are rolled back.
Safe sequence is the sequence in which the processes can be safely executed.

-> Few **print functions** are used for printing the need, allocated, available vector.

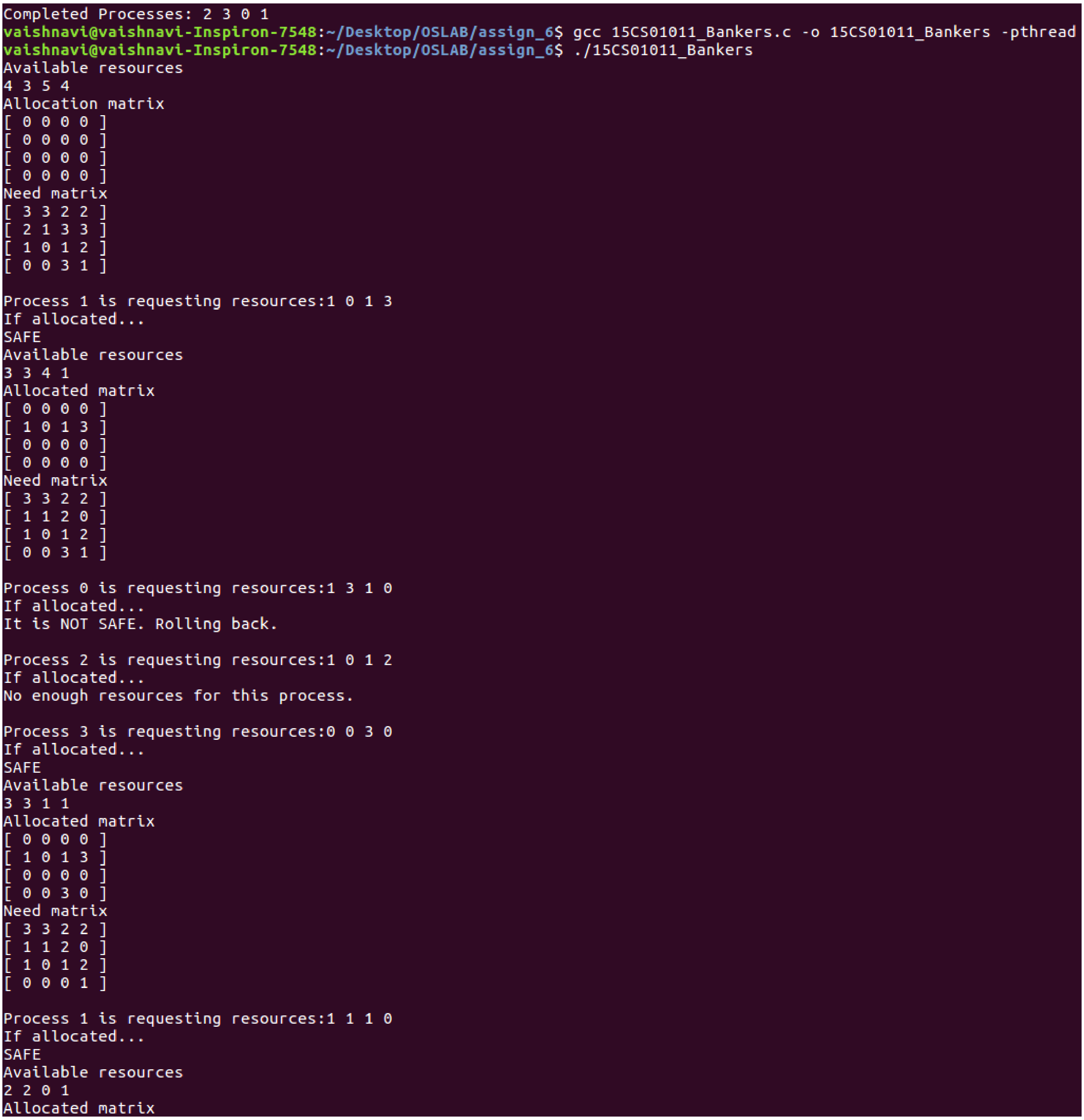
Deadlock Avoidance: As we can see, there is no deadlock in this program. The program terminates after a finite exection time. There is a sequence of completed processes at the end. Therefore, this doesn't contain deadlock.

Compilation: gcc 15CS01011_Bankers.c -o 15CS01011_Bankers -pthread

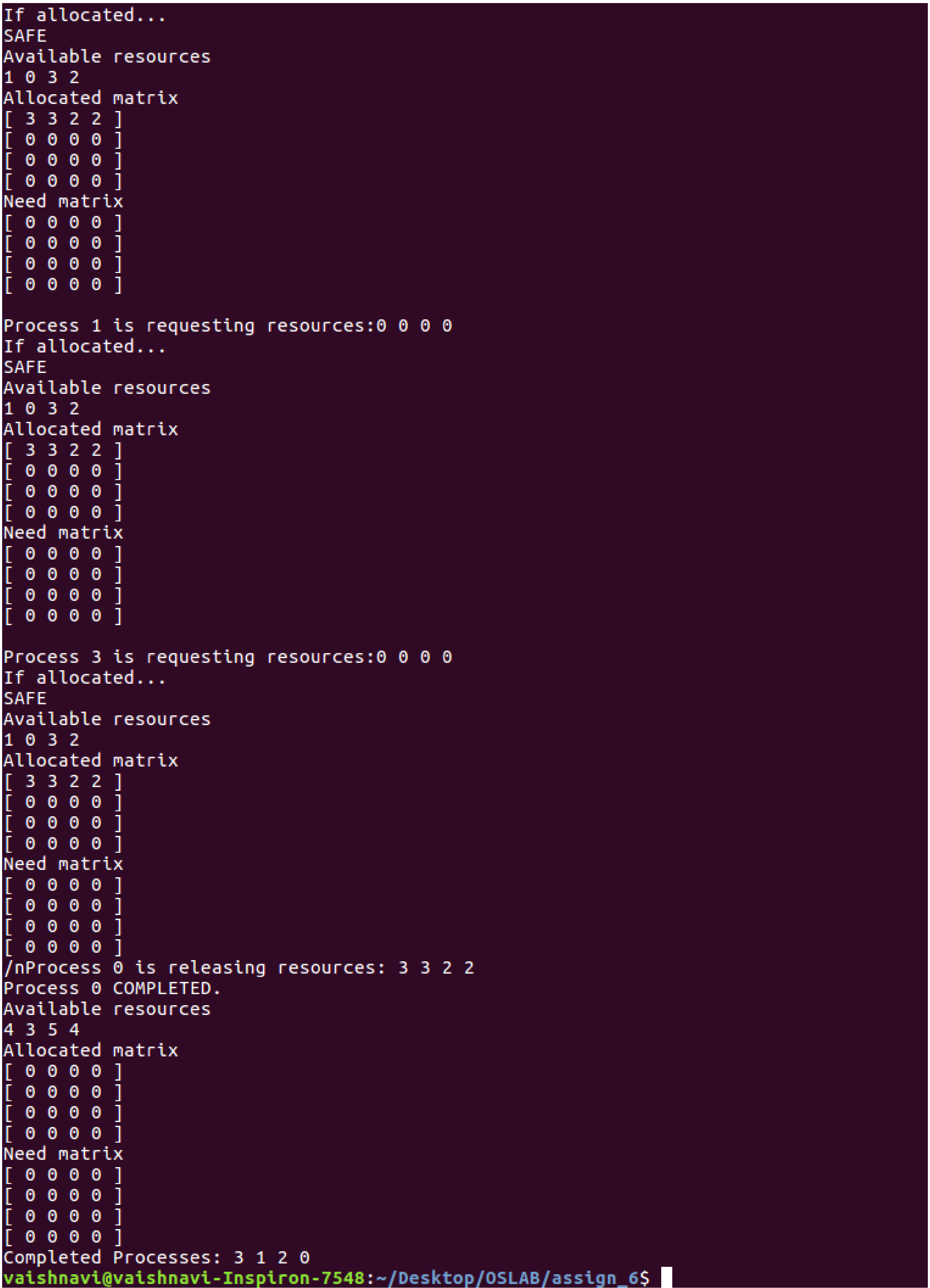
Execution: ./15CS01011_Bankers

Explanation – Initially, the resources available and the need and allocated matrices are as seen in the screenshot, then when a request comes. Here in this request has come from Process 1 as [1 0 1 3]. It checks with the test_safety and finds that there exists a safe sequence. So, it allocates these resources to Process 1 and continues.....

At the end, we can see that all processes terminates with the completion order as 3 -> 1 -> 2 -> 0.



```
Completed Processes: 2 3 0 1
vaishnavi@vaishnavi-Inspiron-7548:~/Desktop/OSLAB/assign_6$ gcc 15CS01011_Bankers.c -o 15CS01011_Bankers -pthread
vaishnavi@vaishnavi-Inspiron-7548:~/Desktop/OSLAB/assign_6$ ./15CS01011_Bankers
Available resources
4 3 5 4
Allocation matrix
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Need matrix
[ 3 3 2 2 ]
[ 2 1 3 3 ]
[ 1 0 1 2 ]
[ 0 0 3 1 ]

Process 1 is requesting resources:1 0 1 3
If allocated...
SAFE
Available resources
3 3 4 1
Allocated matrix
[ 0 0 0 0 ]
[ 1 0 1 3 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Need matrix
[ 3 3 2 2 ]
[ 1 1 2 0 ]
[ 1 0 1 2 ]
[ 0 0 3 1 ]

Process 0 is requesting resources:1 3 1 0
If allocated...
It is NOT SAFE. Rolling back.

Process 2 is requesting resources:1 0 1 2
If allocated...
No enough resources for this process.

Process 3 is requesting resources:0 0 3 0
If allocated...
SAFE
Available resources
3 3 1 1
Allocated matrix
[ 0 0 0 0 ]
[ 1 0 1 3 ]
[ 0 0 0 0 ]
[ 0 0 3 0 ]
Need matrix
[ 3 3 2 2 ]
[ 1 1 2 0 ]
[ 1 0 1 2 ]
[ 0 0 0 1 ]

Process 1 is requesting resources:1 1 1 0
If allocated...
SAFE
Available resources
2 2 0 1
Allocated matrix
```



```
If allocated...
SAFE
Available resources
1 0 3 2
Allocated matrix
[ 3 3 2 2 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Need matrix
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]

Process 1 is requesting resources:0 0 0 0
If allocated...
SAFE
Available resources
1 0 3 2
Allocated matrix
[ 3 3 2 2 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Need matrix
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]

Process 3 is requesting resources:0 0 0 0
If allocated...
SAFE
Available resources
1 0 3 2
Allocated matrix
[ 3 3 2 2 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Need matrix
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
/nProcess 0 is releasing resources: 3 3 2 2
Process 0 COMPLETED.
Available resources
4 3 5 4
Allocated matrix
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Need matrix
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
Completed Processes: 3 1 2 0
vaishnavi@vaishnavi-Inspiron-7548:~/Desktop/OSLAB/assign_6$
```