OS LAB 7

Synchronization

Submitted by : Ridhima Kohli
B19CSE071

How to run files

Type

```
g++ q1.c -o a -lpthread
```

```
./a
```

```
g++ q2.cpp -o a
```

```
./a
```

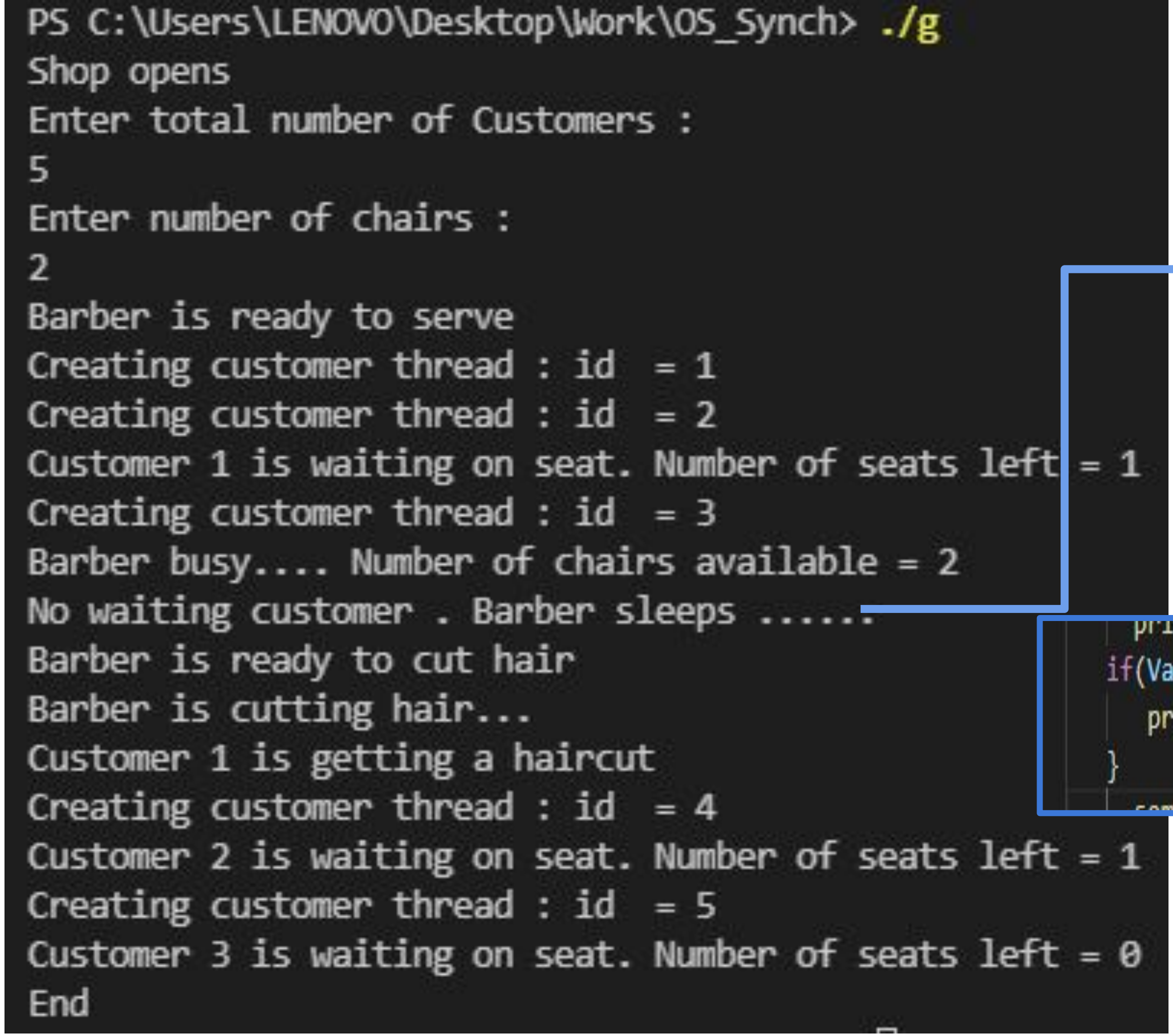
File q1.cpp contains solution to question 1 : Barber

File q2.cpp contains solution to question 2 : Bankers algorithm

Barber Problem

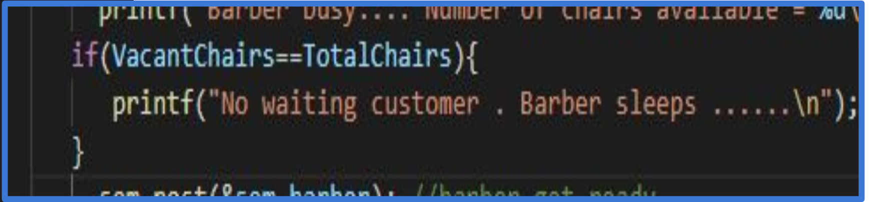
For synchronization we need to use monitors according to question. Since monitors by default are not in C (they are present in Java) so we simulate monitor with struct containing semaphores and required functions.

We have created a barber thread and various customer threads which synchronize and according to availability of seats the output is printed

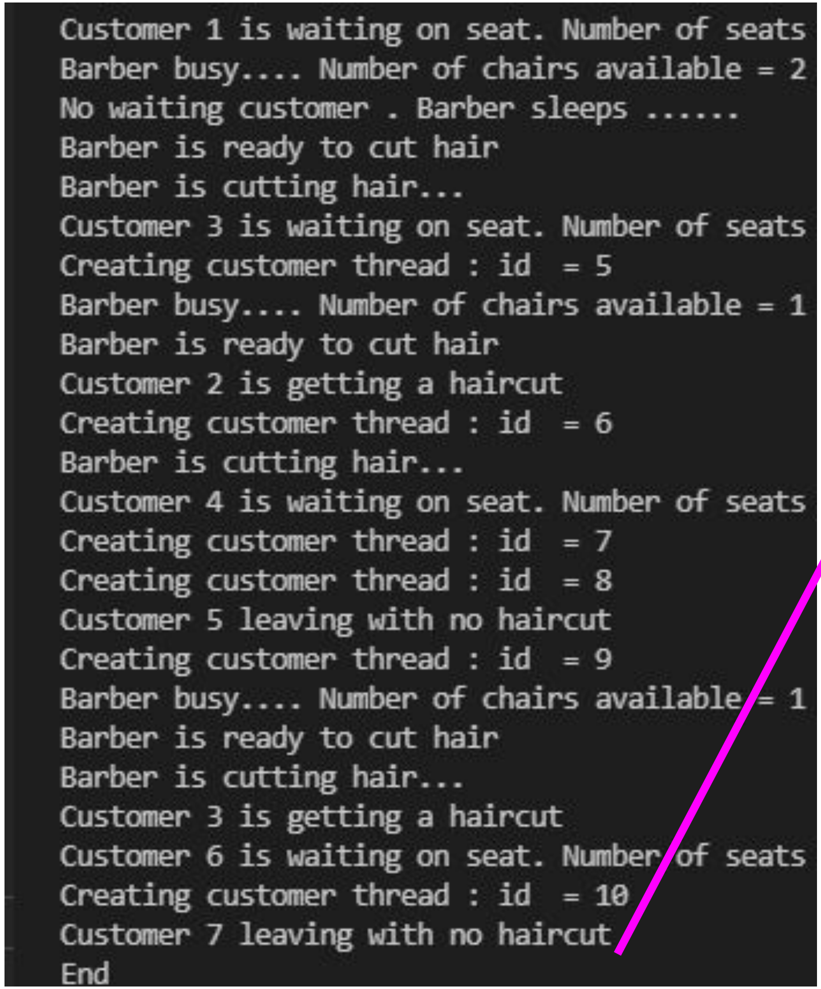


```
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> ./g
Shop opens
Enter total number of Customers :
5
Enter number of chairs :
2
Barber is ready to serve
Creating customer thread : id  = 1
Creating customer thread : id  = 2
Customer 1 is waiting on seat. Number of seats left = 1
Creating customer thread : id  = 3
Barber busy.... Number of chairs available = 2
No waiting customer . Barber sleeps ......
Barber is ready to cut hair
Barber is cutting hair...
Customer 1 is getting a haircut
Creating customer thread : id  = 4
Customer 2 is waiting on seat. Number of seats left = 1
Creating customer thread : id  = 5
Customer 3 is waiting on seat. Number of seats left = 0
End
```

Barber sleeps when no customer

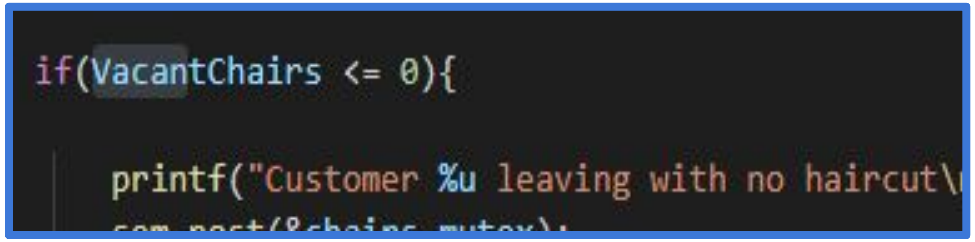


```
 printf("Barber busy.... Number of chairs available = %d\
if(VacantChairs==TotalChairs){
  printf("No waiting customer . Barber sleeps ......\n");
}
 sem_post(&sem_barber); //barber get ready
```



```
Customer 1 is waiting on seat. Number of seats
Barber busy.... Number of chairs available = 2
No waiting customer . Barber sleeps ......
Barber is ready to cut hair
Barber is cutting hair...
Customer 3 is waiting on seat. Number of seats
Creating customer thread : id  = 5
Barber busy.... Number of chairs available = 1
Barber is ready to cut hair
Customer 2 is getting a haircut
Creating customer thread : id  = 6
Barber is cutting hair...
Customer 4 is waiting on seat. Number of seats
Creating customer thread : id  = 7
Creating customer thread : id  = 8
Customer 5 leaving with no haircut
Creating customer thread : id  = 9
Barber busy.... Number of chairs available = 1
Barber is ready to cut hair
Barber is cutting hair...
Customer 3 is getting a haircut
Customer 6 is waiting on seat. Number of seats
Creating customer thread : id  = 10
Customer 7 leaving with no haircut
End
```

Customer leaves without haircut when empty seats are equal to zero



```
if(VacantChairs <= 0){

    printf("Customer %u leaving with no haircut\
```

Q2 part 2 : Bankers algorithm and resource request

Requirements of question :

Input the number of processes, number of resource type and the matrices (Available, Max, Allocation) , a process request (process no. and a request string depicting the number of instances required for each resource type).

Output : Print whether state is safe / unsafe

Print whether request can be served or not

Implementation :

Algorithm for need matrix calculation

Algorithm for checking whether state is safe or unsafe

If state is safe , we have also printed the SAFE sequence

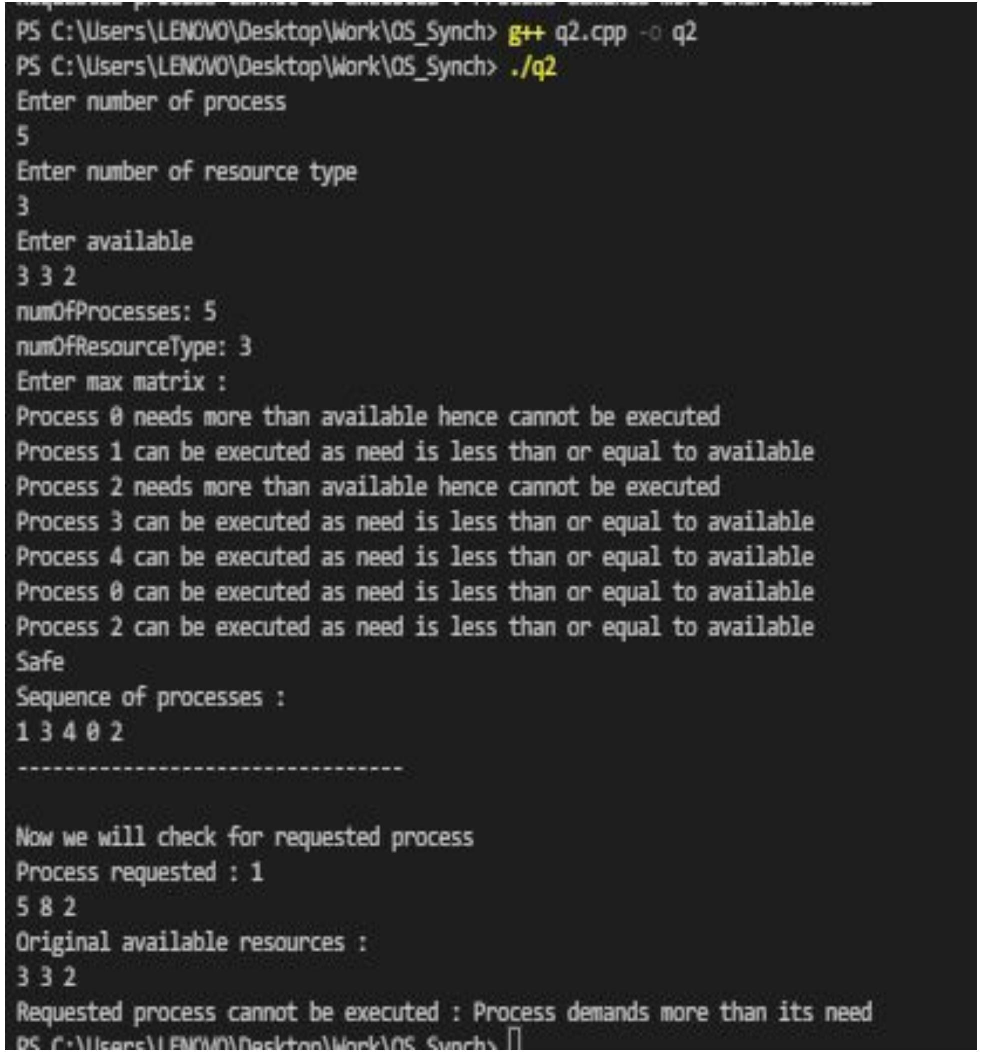
Algorithm for checking whether request can be served or not based on availability input and need matrix comparison

If request instance > need then process cant be granted

Else if availability > request instance then process can be granted

Else if program is in unsafe state then its already deadlock request cant be granted

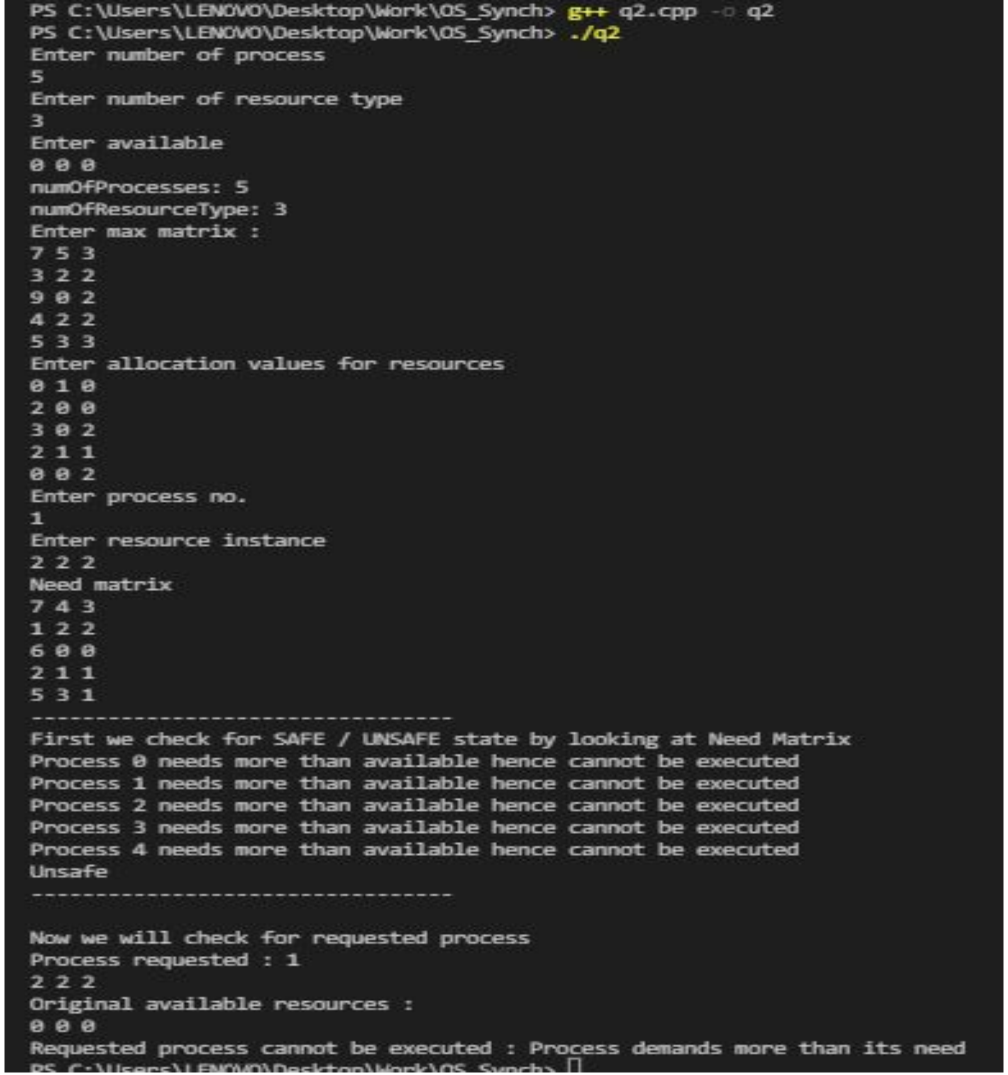
Else we need to check whether availability after other processes > request then request can be granted



```
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> g++ q2.cpp -o q2
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> ./q2
Enter number of process
5
Enter number of resource type
3
Enter available
3 3 2
numOfProcesses: 5
numOfResourceType: 3
Enter max matrix :
Process 0 needs more than available hence cannot be executed
Process 1 can be executed as need is less than or equal to available
Process 2 needs more than available hence cannot be executed
Process 3 can be executed as need is less than or equal to available
Process 4 can be executed as need is less than or equal to available
Process 0 can be executed as need is less than or equal to available
Process 2 can be executed as need is less than or equal to available
Safe
Sequence of processes :
1 3 4 0 2
--------------------------------

Now we will check for requested process
Process requested : 1
5 8 2
Original available resources :
3 3 2
Requested process cannot be executed : Process demands more than its need
```

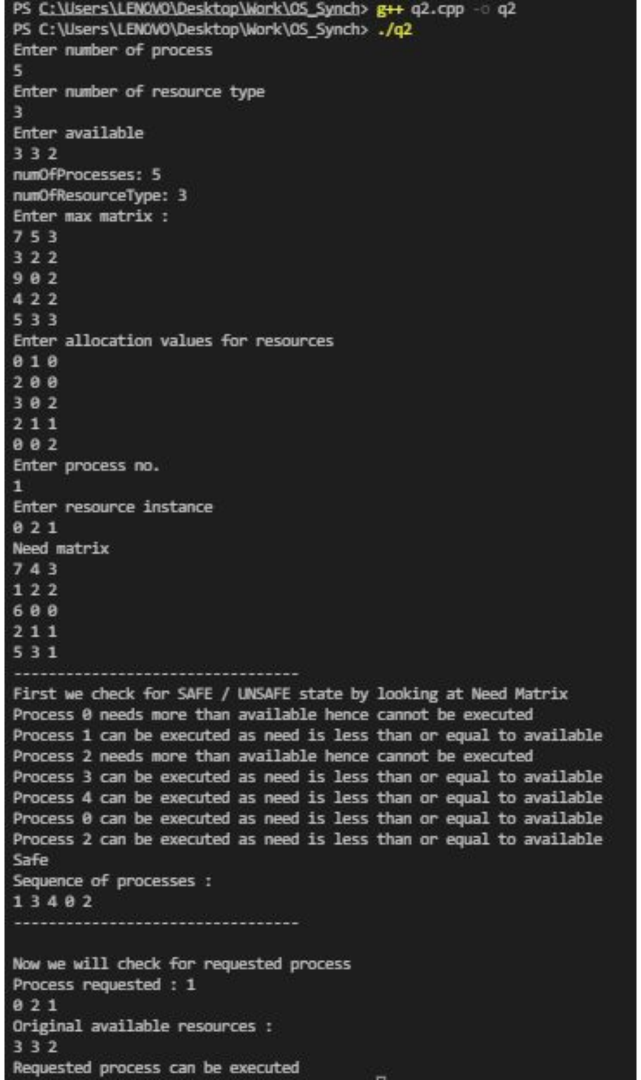
Safe
Resource not granted to request



```
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> g++ q2.cpp -o q2
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> ./q2
Enter number of process
5
Enter number of resource type
3
Enter available
0 0 0
numOfProcesses: 5
numOfResourceType: 3
Enter max matrix :
7 5 3
3 2 2
9 0 2
4 2 2
5 3 3
Enter allocation values for resources
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter process no.
1
Enter resource instance
2 2 2
Need matrix
7 4 3
1 2 2
6 0 0
2 1 1
5 3 1
---------------------------------
First we check for SAFE / UNSAFE state by looking at Need Matrix
Process 0 needs more than available hence cannot be executed
Process 1 needs more than available hence cannot be executed
Process 2 needs more than available hence cannot be executed
Process 3 needs more than available hence cannot be executed
Process 4 needs more than available hence cannot be executed
Unsafe
---------------------------------

Now we will check for requested process
Process requested : 1
2 2 2
Original available resources :
0 0 0
Requested process cannot be executed : Process demands more than its need
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> 
```

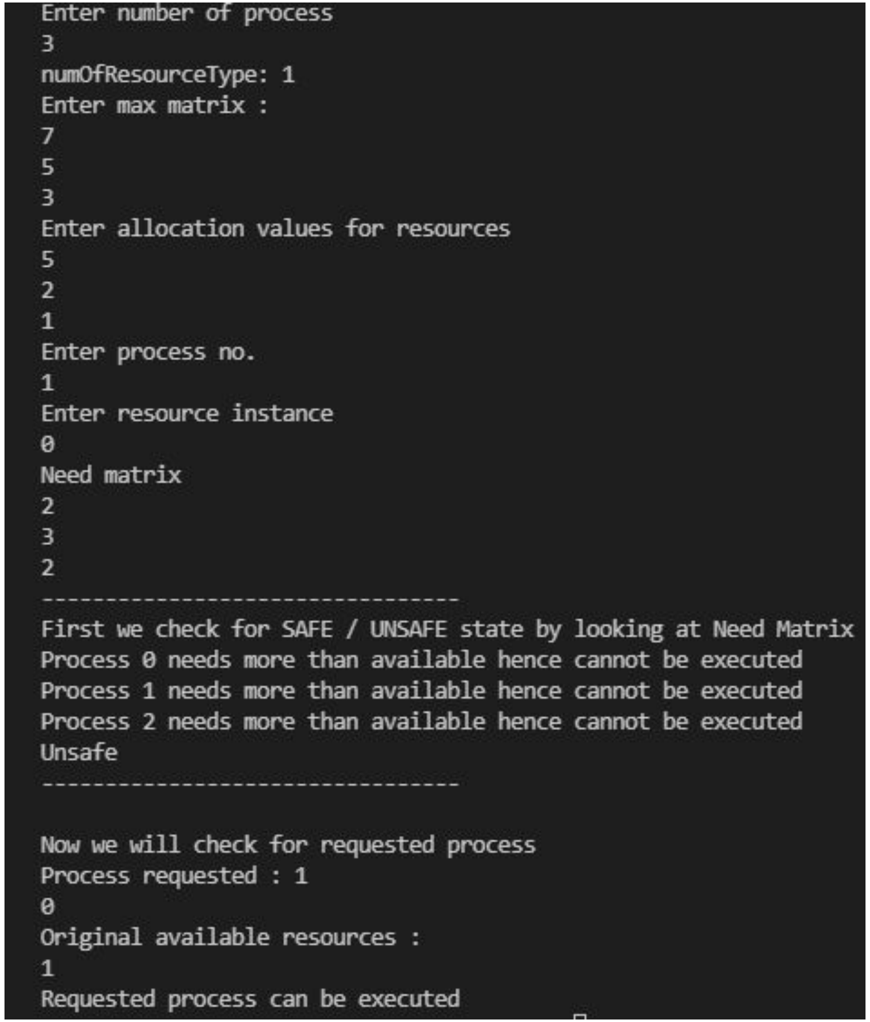
Unsafe
Resource not granted to request



```
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> g++ q2.cpp -o q2
PS C:\Users\LENOVO\Desktop\Work\OS_Synch> ./q2
Enter number of process
5
Enter number of resource type
3
Enter available
3 3 2
numOfProcesses: 5
numOfResourceType: 3
Enter max matrix :
7 5 3
3 2 2
9 0 2
4 2 2
5 3 3
Enter allocation values for resources
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter process no.
1
Enter resource instance
0 2 1
Need matrix
7 4 3
1 2 2
6 0 0
2 1 1
5 3 1
---------------------------------
First we check for SAFE / UNSAFE state by looking at Need Matrix
Process 0 needs more than available hence cannot be executed
Process 1 can be executed as need is less than or equal to available
Process 2 needs more than available hence cannot be executed
Process 3 can be executed as need is less than or equal to available
Process 4 can be executed as need is less than or equal to available
Process 0 can be executed as need is less than or equal to available
Process 2 can be executed as need is less than or equal to available
Safe
Sequence of processes :
1 3 4 0 2
---------------------------------

Now we will check for requested process
Process requested : 1
0 2 1
Original available resources :
3 3 2
Requested process can be executed
```

Safe
Resource granted to request



```
Enter number of process
3
numOfResourceType: 1
Enter max matrix :
7
5
3
Enter allocation values for resources
5
2
1
Enter process no.
1
Enter resource instance
0
Need matrix
2
3
2
--------------------------------
First we check for SAFE / UNSAFE state by looking at Need Matrix
Process 0 needs more than available hence cannot be executed
Process 1 needs more than available hence cannot be executed
Process 2 needs more than available hence cannot be executed
Unsafe
--------------------------------

Now we will check for requested process
Process requested : 1
0
Original available resources :
1
Requested process can be executed
```

Unsafe
Request granted