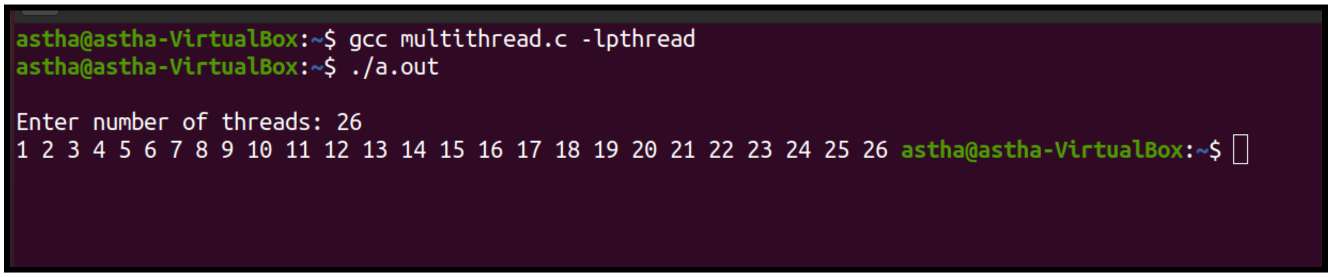
Astha Patel - AU1940312
CSE-332 : Operating System
Section-2
End sem Lab exam - Description

Question:1 (b)

Screenshots



```
astha@astha-VirtualBox:~$ gcc multithread.c -lpthread
astha@astha-VirtualBox:~$ ./a.out

Enter number of threads: 26
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 astha@astha-VirtualBox:~$ 
```

Explanation:

- Over here, multi threads are used to print the numbers using thread synchronization.
- Threads are the light weight processes and they are not independent of each other like processes.
- Using the pthread library, for the first thread count, 1 is printed . For the second thread count, 2 is printed and so on.

Question:2

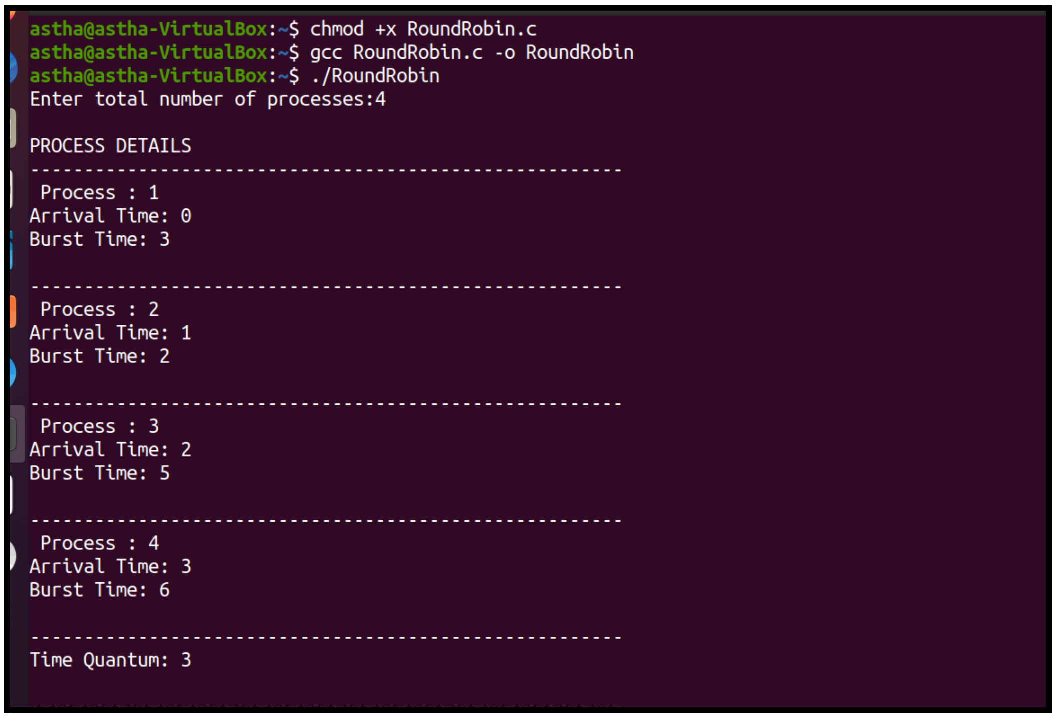
a. Describe the RoundRobin (RR) and Modified RoundRobin (MRR) Algorithm. Also mention the difference between the results of both the algorithms and implement them both in C.

Round Robin

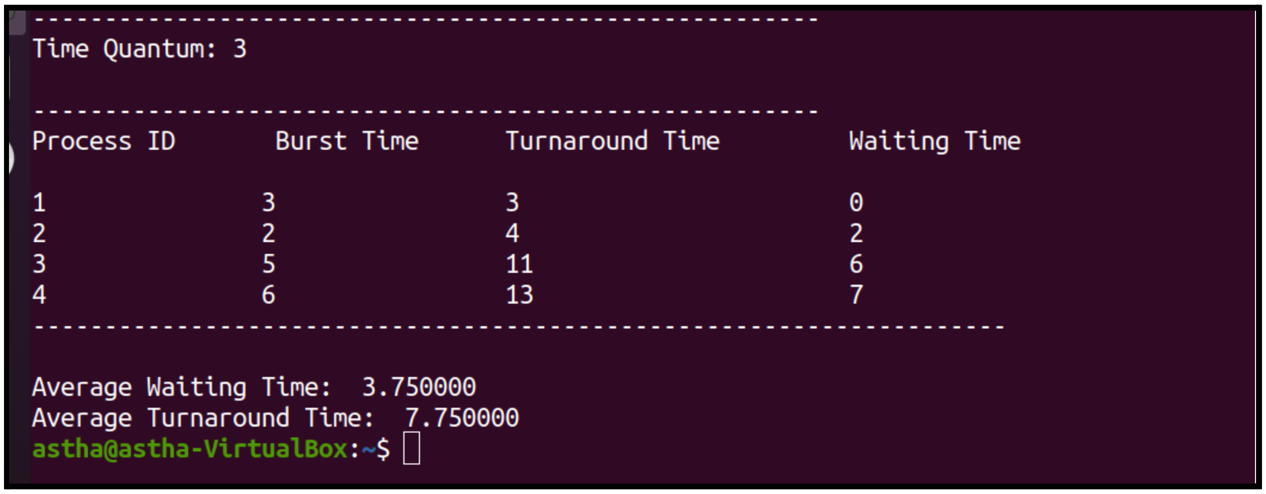
- Round robin algorithm is the algorithm which is used by the CPU for process scheduling in such a way that the CPU is utilized more.
- Round robin is a preemptive type of algorithm in which a particular time (limited amount of time) is allocated to the CPU for a process to schedule which is the time slice between each request which is called time quantum.

- This scheduler saves the progress of the job during the context switch as CPU cannot hold a process for a long time.
- The efficiency over here depends on the quantum time value.

Screenshots



```
astha@astha-VirtualBox:~$ chmod +x RoundRobin.c
astha@astha-VirtualBox:~$ gcc RoundRobin.c -o RoundRobin
astha@astha-VirtualBox:~$ ./RoundRobin
Enter total number of processes:4

PROCESS DETAILS
-------------------------------------------------------
 Process : 1
Arrival Time: 0
Burst Time: 3

-------------------------------------------------------
 Process : 2
Arrival Time: 1
Burst Time: 2

-------------------------------------------------------
 Process : 3
Arrival Time: 2
Burst Time: 5

-------------------------------------------------------
 Process : 4
Arrival Time: 3
Burst Time: 6

-------------------------------------------------------
Time Quantum: 3
```



```
-------------------------------------------------------
Time Quantum: 3

-------------------------------------------------------
Process ID       Burst Time      Turnaround Time         Waiting Time

1                3               3                       0
2                2               4                       2
3                5               11                      6
4                6               13                      7
--------------------------------------------------------------------

Average Waiting Time:  3.750000
Average Turnaround Time:   7.750000
astha@astha-VirtualBox:~$ 
```

Explanation:

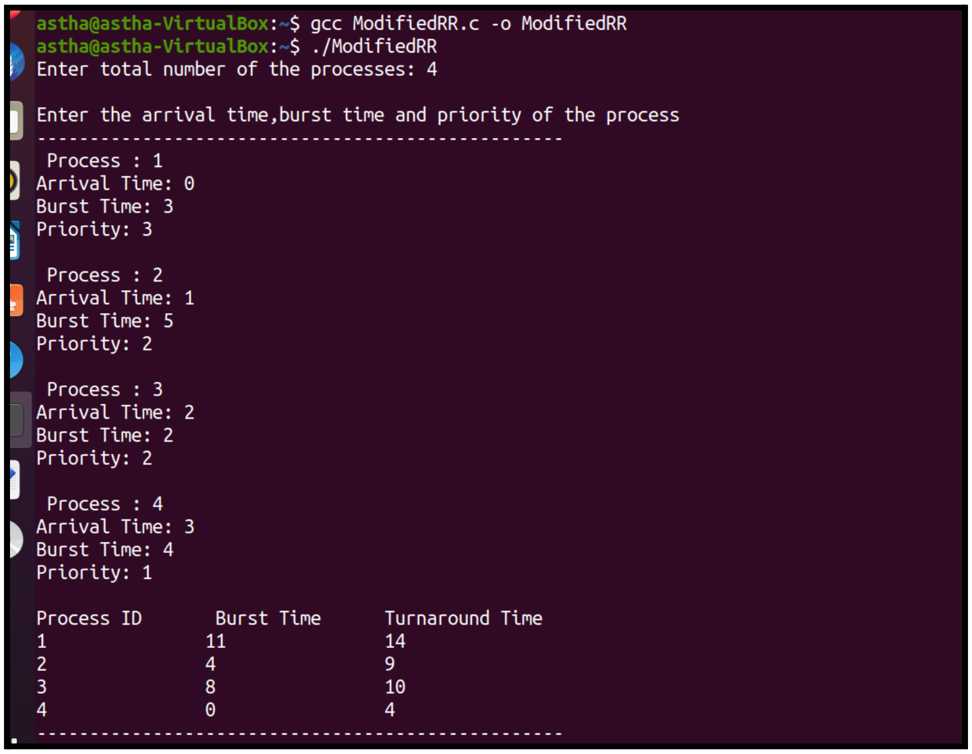
- As you can see in the output, firstly the user is asked to enter the total number of processes and their arrival, burst time and time quantum.
- The processes are arranged in first come first order in the queue.
- The quantum value is allocated to each process and the first process is executed till the time quantum after that the context switching happens and the state of the first process is saved.
- Like this, all the steps are repeated till the number of processes are over.
- The turnaround time and the waiting time are the main components to be calculated.
- They are calculated using the round robin algorithm.

  Waiting time = Arrival time - Burst time
  Turnaround time = Total time - Waiting time

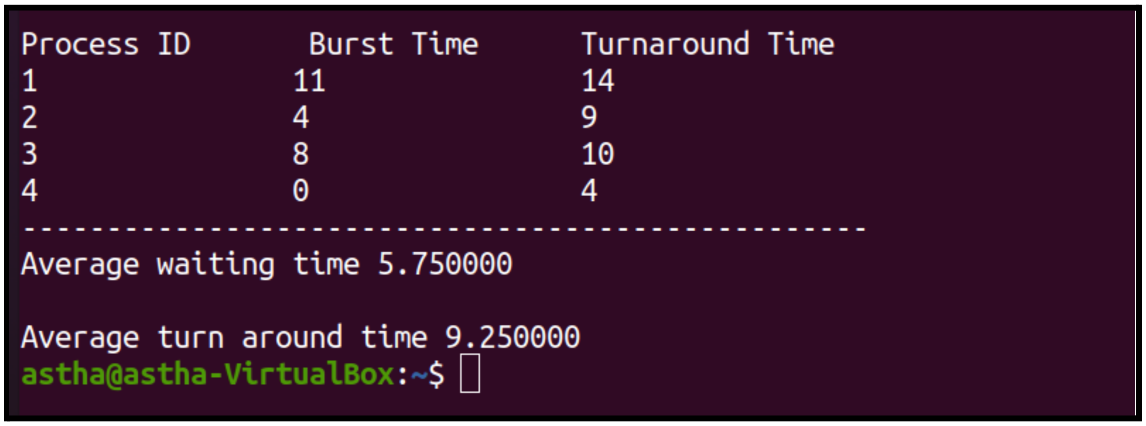
Modified Round Robin:

- Modified RR is the same as the round robin scheduling but the difference is only the priority is given to the process in modified RR.
- Thus, now the priority given to a process is considered to be the first step of this algorithm.

Screenshots

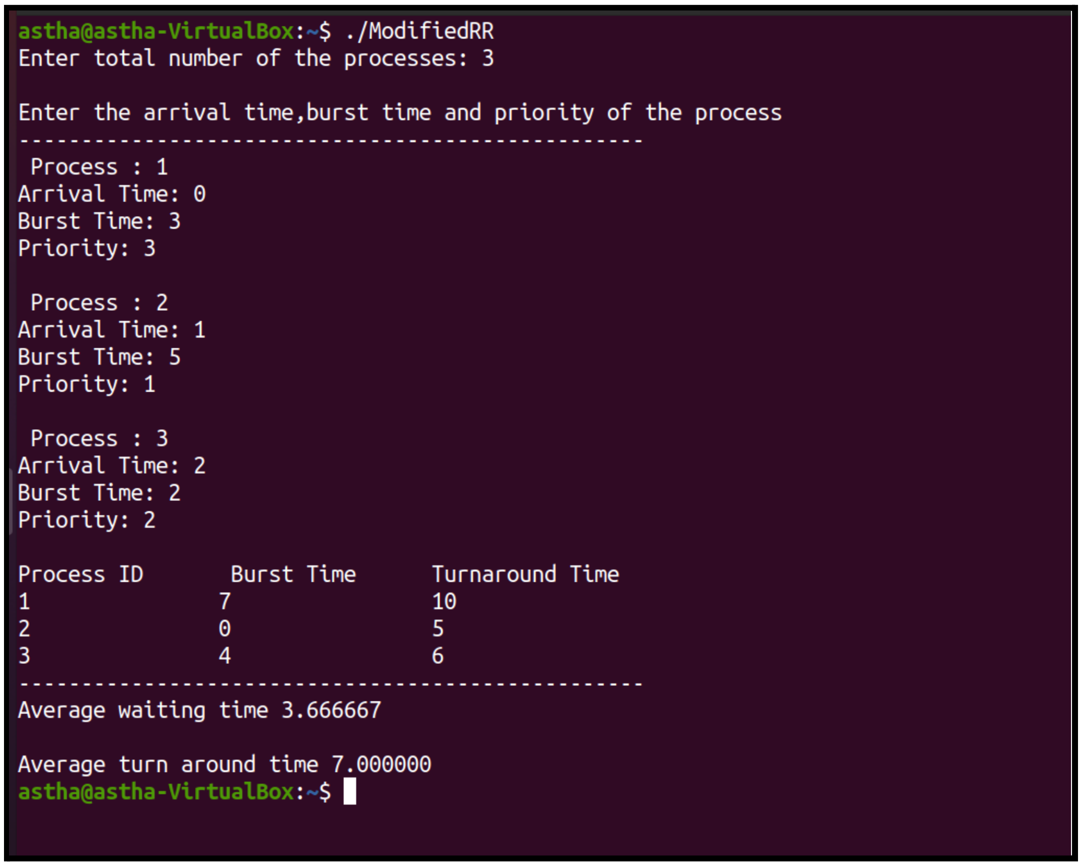


```
astha@astha-VirtualBox:~$ gcc ModifiedRR.c -o ModifiedRR
astha@astha-VirtualBox:~$ ./ModifiedRR
Enter total number of the processes: 4

Enter the arrival time,burst time and priority of the process
--------------------------------------------------
 Process : 1
Arrival Time: 0
Burst Time: 3
Priority: 3

 Process : 2
Arrival Time: 1
Burst Time: 5
Priority: 2

 Process : 3
Arrival Time: 2
Burst Time: 2
Priority: 2

 Process : 4
Arrival Time: 3
Burst Time: 4
Priority: 1

Process ID       Burst Time      Turnaround Time
1               11              14
2               4               9
3               8               10
4               0               4
--------------------------------------------------
```

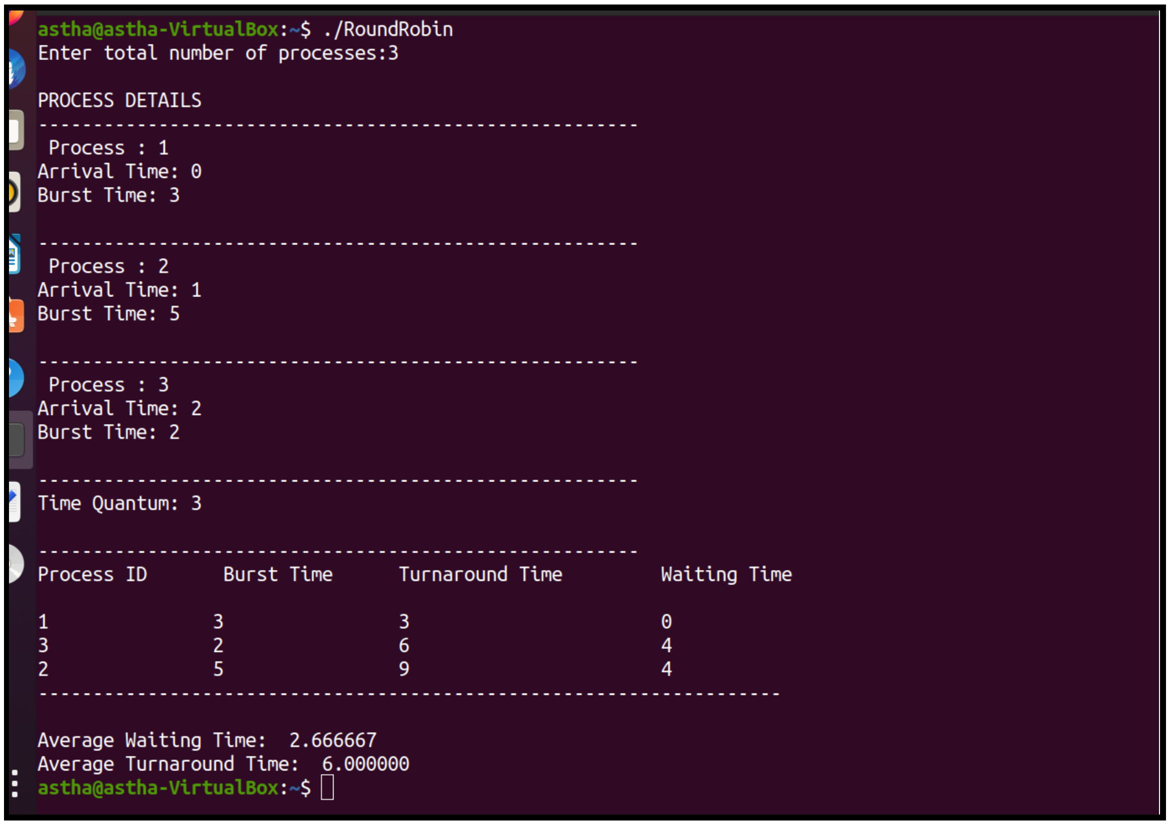


```
Process ID        Burst Time       Turnaround Time
1                11               14
2                4                9
3                8                10
4                0                4
--------------------------------------------------
Average waiting time 5.750000

Average turn around time 9.250000
astha@astha-VirtualBox:~$ 
```

Comparing the outputs of both having same number of process , same arrival time and burst time



```
astha@astha-VirtualBox:~$ ./ModifiedRR
Enter total number of the processes: 3

Enter the arrival time,burst time and priority of the process
--------------------------------------------------
 Process : 1
Arrival Time: 0
Burst Time: 3
Priority: 3

 Process : 2
Arrival Time: 1
Burst Time: 5
Priority: 1

 Process : 3
Arrival Time: 2
Burst Time: 2
Priority: 2

Process ID       Burst Time      Turnaround Time
1               7               10
2               0               5
3               4               6
--------------------------------------------------
Average waiting time 3.666667

Average turn around time 7.000000
astha@astha-VirtualBox:~$ 
```



```
astha@astha-VirtualBox:~$ ./RoundRobin
Enter total number of processes:3

PROCESS DETAILS
-------------------------------------------------------
 Process : 1
Arrival Time: 0
Burst Time: 3

-------------------------------------------------------
 Process : 2
Arrival Time: 1
Burst Time: 5

-------------------------------------------------------
 Process : 3
Arrival Time: 2
Burst Time: 2

-------------------------------------------------------
Time Quantum: 3

-------------------------------------------------------
Process ID       Burst Time      Turnaround Time        Waiting Time

1                3               3                      0
3                2               6                      4
2                5               9                      4
--------------------------------------------------------------------

Average Waiting Time:  2.666667
Average Turnaround Time:  6.000000
astha@astha-VirtualBox:~$ 
```

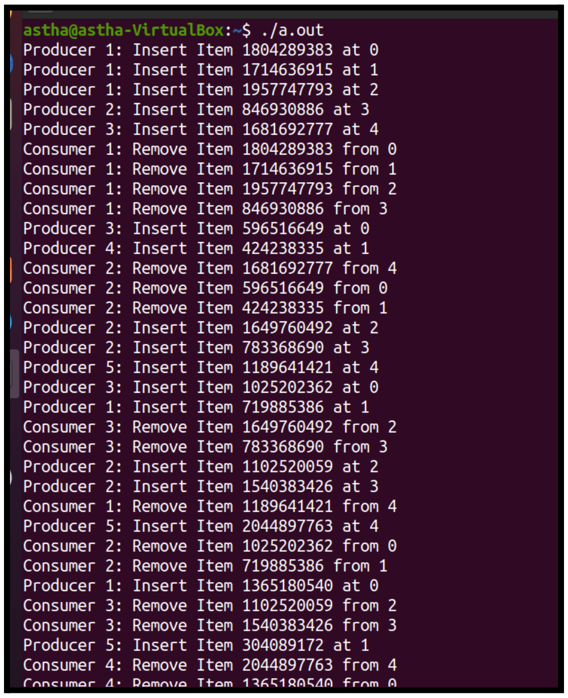
- As the time quantum is static the CPU's performance decreases.
- The round robin algorithm does not use the priority while scheduling, thus the CPU performance decreases. This problem is solved in the Modified RR algorithm.
- The Modified Round Robin algorithm generates the ready queue by considering the priority of the processes, which reduces the turnaround time and waiting time and the context switching between the processes.
- Thus, the CPU performance increases in the Modified RR algorithm.

Question:3

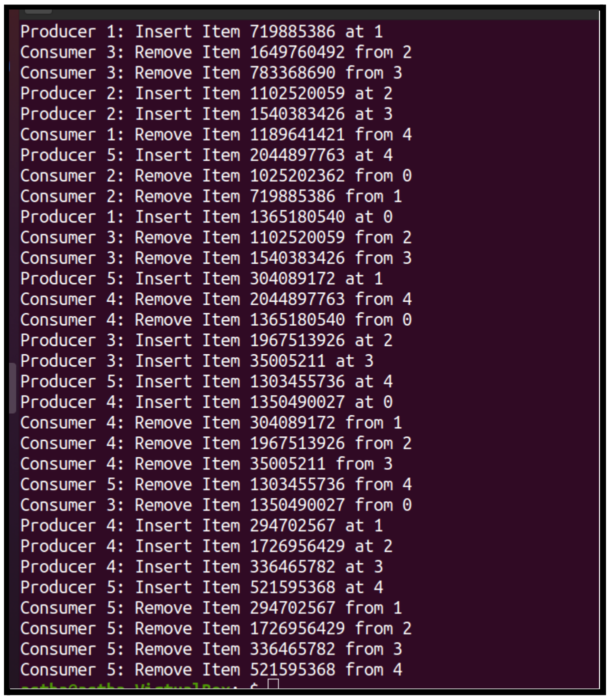
Describe what is the Producer Consumer Problem and its solution in detail using Semaphores and Mutex and implement it in C.

- Producer and consumer problem is an example of the multi-process synchronization in which they share a fixed size buffer which is used by the queue.
- The job of the producer is to generate the data and store it into the buffer.
- The job of the consumer is to consume the data (removing it from the buffer) from this buffer simultaneously.

Screenshots



```
astha@astha-VirtualBox:~$ ./a.out
Producer 1: Insert Item 1804289383 at 0
Producer 1: Insert Item 1714636915 at 1
Producer 1: Insert Item 1957747793 at 2
Producer 2: Insert Item 846930886 at 3
Producer 3: Insert Item 1681692777 at 4
Consumer 1: Remove Item 1804289383 from 0
Consumer 1: Remove Item 1714636915 from 1
Consumer 1: Remove Item 1957747793 from 2
Consumer 1: Remove Item 846930886 from 3
Producer 3: Insert Item 596516649 at 0
Producer 4: Insert Item 424238335 at 1
Consumer 2: Remove Item 1681692777 from 4
Consumer 2: Remove Item 596516649 from 0
Consumer 2: Remove Item 424238335 from 1
Producer 2: Insert Item 1649760492 at 2
Producer 2: Insert Item 783368690 at 3
Producer 5: Insert Item 1189641421 at 4
Producer 3: Insert Item 1025202362 at 0
Producer 1: Insert Item 719885386 at 1
Consumer 3: Remove Item 1649760492 from 2
Consumer 3: Remove Item 783368690 from 3
Producer 2: Insert Item 1102520059 at 2
Producer 2: Insert Item 1540383426 at 3
Consumer 1: Remove Item 1189641421 from 4
Producer 5: Insert Item 2044897763 at 4
Consumer 2: Remove Item 1025202362 from 0
Consumer 2: Remove Item 719885386 from 1
Producer 1: Insert Item 1365180540 at 0
Consumer 3: Remove Item 1102520059 from 2
Consumer 3: Remove Item 1540383426 from 3
Producer 5: Insert Item 304089172 at 1
Consumer 4: Remove Item 2044897763 from 4
Consumer 4: Remove Item 1365180540 from 0
```



```
Producer 1: Insert Item 719885386 at 1
Consumer 3: Remove Item 1649760492 from 2
Consumer 3: Remove Item 783368690 from 3
Producer 2: Insert Item 1102520059 at 2
Producer 2: Insert Item 1540383426 at 3
Consumer 1: Remove Item 1189641421 from 4
Producer 5: Insert Item 2044897763 at 4
Consumer 2: Remove Item 1025202362 from 0
Consumer 2: Remove Item 719885386 from 1
Producer 1: Insert Item 1365180540 at 0
Consumer 3: Remove Item 1102520059 from 2
Consumer 3: Remove Item 1540383426 from 3
Producer 5: Insert Item 304089172 at 1
Consumer 4: Remove Item 2044897763 from 4
Consumer 4: Remove Item 1365180540 from 0
Producer 3: Insert Item 1967513926 at 2
Producer 3: Insert Item 35005211 at 3
Producer 5: Insert Item 1303455736 at 4
Producer 4: Insert Item 1350490027 at 0
Consumer 4: Remove Item 304089172 from 1
Consumer 4: Remove Item 1967513926 from 2
Consumer 4: Remove Item 35005211 from 3
Consumer 5: Remove Item 1303455736 from 4
Consumer 3: Remove Item 1350490027 from 0
Producer 4: Insert Item 294702567 at 1
Producer 4: Insert Item 1726956429 at 2
Producer 4: Insert Item 336465782 at 3
Producer 5: Insert Item 521595368 at 4
Consumer 5: Remove Item 294702567 from 1
Consumer 5: Remove Item 1726956429 from 2
Consumer 5: Remove Item 336465782 from 3
Consumer 5: Remove Item 521595368 from 4
astha@astha-VirtualBox:~$
```

Explanation :

- This program is coded using the mutex and semaphore for the producer and consumer solution.
- I have used 5 producers and 5 consumers in this solution (in the above screenshot).
- If the buffer is full, the producer goes to sleep or discards the data.
- The consumer consumes the data from the buffer notifying the producer to start filling the buffer again.
- The same thing happens with the consumer when it finds the buffer empty.
- The consumer goes to sleep till the producer puts the data into the buffer and notifies the consumer to wake up from the sleep as the buffer is not empty.