EX NO: 5	**SEMAPHORES**
DATE:	

AIM:

To write a C program to implement the Producer & consumer Problem (Semaphore).

ALGORITHM:

1: The Semaphore mutex, full & empty are initialized.

2: In the case of producer process

i) Produce an item in to temporary variable.

ii) If there is empty space in the buffer check the mutex value for enter into the critical section.

iii)If the mutex value is 0, allow the producer to add value in the temporary variable to the buffer.

3: In the case of consumer process

i) It should wait if the buffer is empty ii) If there is any item in the buffer check for mutex value, if the mutex==0, remove item from buffer iii) Signal the mutex value and reduce the empty value by 1.

iv) Consume the item.

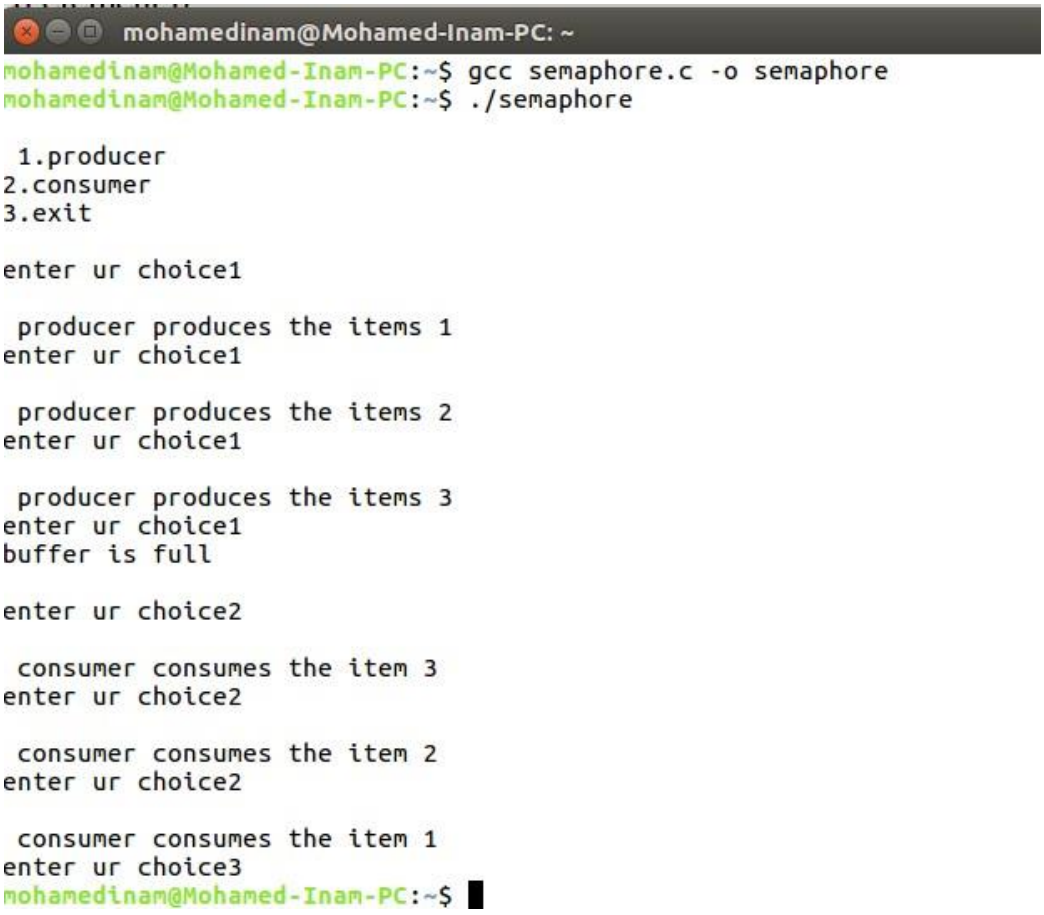
4: Print the result

PROGRAM

```
#include<stdio.h>
int mutex=1,full=0,empty=3,x=0;
main()
{
int n;
void producer();
void consumer();
int wait(int);
int signal(int);
printf("\n 1.producer\n2.consumer\n3.exit\n");
while(1) {
printf(" \nenter ur choice");
scanf("%d",&n);
switch(n)
{
case 1:if((mutex==1)&&(empty!=0))
producer();
else
printf("buffer is full\n");
break;
case 2:if((mutex==1)&&(full!=0))
consumer();
else
printf("buffer is empty");
break;
case 3:exit(0);
break;
}
}
}
int wait(int s)
{
        return(--s);
}
int signal(int s)
{
        return (++s);
 }
void producer()
{
mutex=wait(mutex);
full=signal(full);
empty=wait(empty);
```

```
 x++;
printf("\n producer produces the items %d",x);
mutex=signal(mutex);
}
void consumer()
{
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\n consumer consumes the item %d",x);
x--;
mutex=signal(mutex);
}
```

OUTPUT:



RESULT:

Thus the program for Producer & consumer Problem using Semaphore was executed successfull