Name: Aditya Somani
Roll No: T1851061
Div: A
PRN NO. 71901204L

ASSIGNMENT NO. 4

TITLE: Thread synchronization using counting semaphores.

Program:

```
#include<stdio.h>
#include<pthread.h>       //Thread library
#include<semaphore.h>//Headerfileforsemaphore

#define BUFF_SIZE 5

typedef intbuffer_t;        //internalBuffer
buffer_tbuff[BUFF_SIZE];
int bindex;

pthread_mutex_tbuff_mutex; //Mutex for buffer

sem_tfull;    //When 0 buffer isfull
sem_tempty;        //When 0 buffer isempty

/*==Inserting item into the buffer==*/

void insert_item(buffer_t item)
{
        if(bindex<BUFF_SIZE)
                buff[bindex++]=item;
```

```
else
        printf("\nBuffer is full...!!!!");
```

```
}

/*==Removing item from the buffer==*/

buffer_tremove_item()
{
        if(bindex>0)
                return(buff[--bindex]);
        else
                printf("\nBuffer is Empty ...!!!!");
}

/*==Check if buffer is empty or not==*/

int isempty()
{
        if(bindex==0)
                return 1;
        else
                return 0;
}

/*==Check if buffer is full or not==*/

int isfull()
{
```

```
        if(bindex==BUFF_SIZE)
                return 1;
        else
                return 0;
}

/*===Producer will produce the elements===*/

void *producer(void *para)
{
        int *thread=(int *)para;
        buffer_tval;
        int i=0;
        sleep(rand()%10); //sleep for random time
        val=rand()%100;             //Produce a randomvalue
        pthread_mutex_lock(&buff_mutex);//lock
        do
        {
                pthread_mutex_unlock(&buff_mutex);//unlock
                sem_wait(&full);            //Decrement
                pthread_mutex_lock(&buff_mutex);//lock
        }while(isfull());
        insert_item(val);    //insert item into the buffer
        pthread_mutex_unlock(&buff_mutex);        //unlock
        sem_post(&empty);        //release semaphore
        printf("\n\tProducer %d produces%d\n",*thread,val);
```

```
        i++;
        pthread_exit(0);
}

/*===Consumer will Consumes the items===*/

void *consumer(void *ptr)
{
        int *thread=(int *)ptr;
        buffer_tval;
        int i=0;

        pthread_mutex_lock(&buff_mutex);
        do
        {
                pthread_mutex_unlock(&buff_mutex);
                sem_wait(&empty);       //increment
                pthread_mutex_lock(&buff_mutex);
        }while(isempty());
        val=remove_item();      //remove item frombuffer
        pthread_mutex_unlock(&buff_mutex);
        sem_post(&full);    //releasesemaphore
        printf("\tConsumer %d Consumed %d from buffer\n",*thread,val);
        i++;

        pthread_exit(0);
```

```
}

int main(int argc,char *argv[])
{
        pthread_mutex_init(&buff_mutex,NULL);     //initializemutex
        sem_init(&full,0,BUFF_SIZE);     //initialize semaphore
        sem_init(&empty,0,0);
        bindex=0;

        pthread_t tid1[10],tid2[10];
        int i;
        int num[5];

        // Create Multiple threads
        for(i=0;i<5;i++)
        {       num[i]=i;
                pthread_create(&tid1[i],NULL,&producer,&num[i]);
                pthread_create(&tid2[i],NULL,&consumer,&num[i]);
        }
        // Join threads
        for(i=0;i<5;i++)
        {
                pthread_join(tid1[i],NULL);
                pthread_join(tid2[i],NULL);
        }
```

```
        // Destroying all semaphores & mutex
        pthread_mutex_destroy(&buff_mutex);
        sem_destroy(&full);
        sem_destroy(&empty);


        return 0;
}
```

Output :