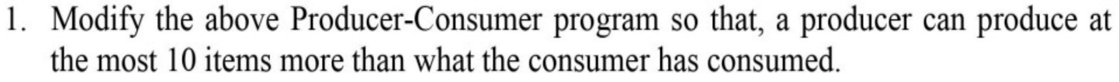
LAB NO 7:

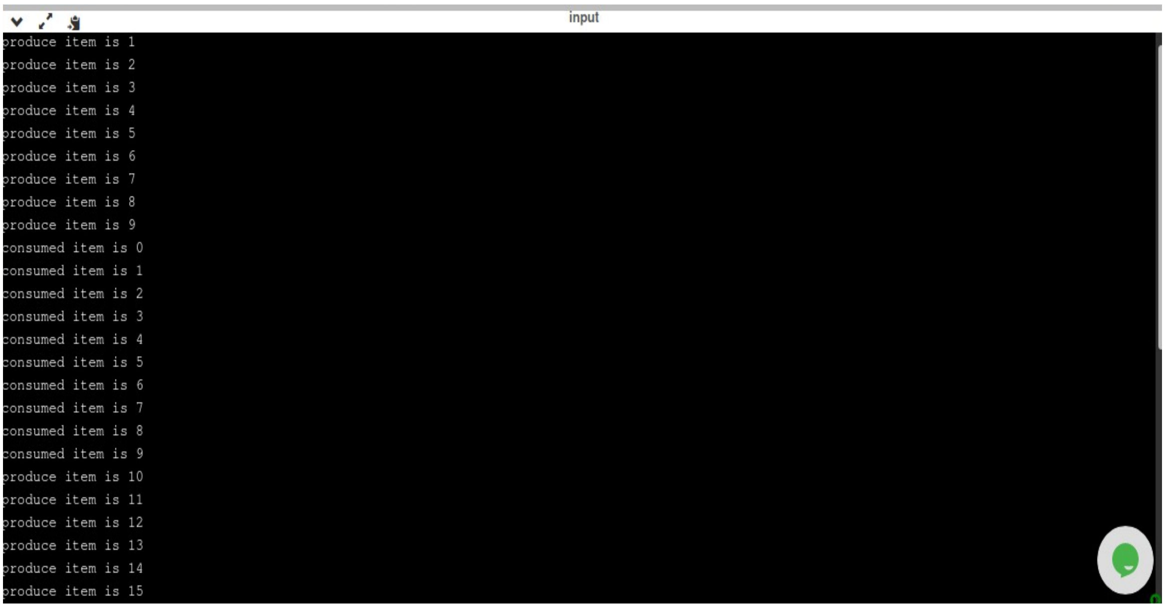


1. Modify the above Producer-Consumer program so that, a producer can produce at the most 10 items more than what the consumer has consumed.

Program :

```
#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<semaphore.h>
int buf[10], f, r;

sem_t mutex, full, empty;

void *produce(void * arg){
        int i;
        for(i = 0; i < 20 ; i++){
                sem_wait(&empty);
                sem_wait(&mutex);
                printf("produce item is %d\n", i);
                buf[(++r)%10] = i;
                sleep(1);
                sem_post(&mutex);
                sem_post(&full);
        }
}

void *consume(void *arg){
        int item, i;
        for(i = 0; i < 20; i++){
                sem_wait(&full);
                sem_wait(&mutex);
                item = buf[(++f)%10];
                printf("consumed item is %d\n", item);
                sleep(1);
                sem_post(&mutex);
                sem_post(&empty);
        }
}

int main(int argc, char const *argv[])
{
        pthread_t tid1, tid2;
        sem_init(&mutex, 0, 1);
        sem_init(&full, 0, 0);
        sem_init(&empty, 0, 10);
        pthread_create(&tid1, NULL, produce, NULL);
        pthread_create(&tid2, NULL, consume, NULL);
        pthread_join(tid1, NULL);
        pthread_join(tid2, NULL);
        return 0;
}
```

Output:



```
produce item is 1
produce item is 2
produce item is 3
produce item is 4
produce item is 5
produce item is 6
produce item is 7
produce item is 8
produce item is 9
consumed item is 0
consumed item is 1
consumed item is 2
consumed item is 3
consumed item is 4
consumed item is 5
consumed item is 6
consumed item is 7
consumed item is 8
consumed item is 9
produce item is 10
produce item is 11
produce item is 12
produce item is 13
produce item is 14
produce item is 15
```

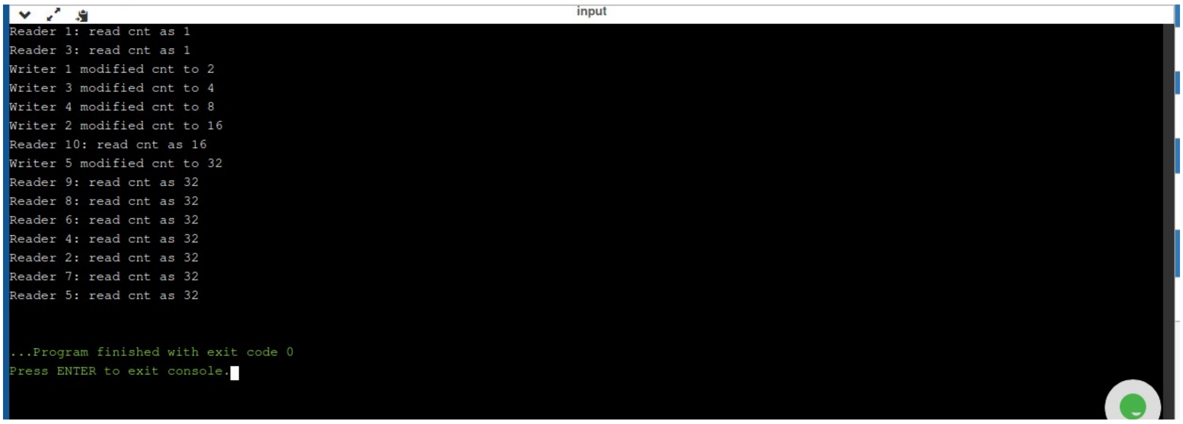
2. Write a C program for the first readers-writers problem using semaphores.

Program :

```
// Reader- writer problem using mutex and sephamore

#include<stdlib.h>
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>

void * reader( void *rno);
void* writer(void *wno);

sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int numreader = 0;


void* reader(void* rno)
{
        //Lock thread before reading
        pthread_mutex_lock(&mutex);
        numreader++;
        if(numreader==1)
        {
```

```
                  // This is the first reader, then it will block writer call
                  sem_wait(&wrt);
            }
            pthread_mutex_unlock(&mutex);
    // Reading Section
    printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);

    // Reader acquire the lock before modifying numreader
    pthread_mutex_lock(&mutex);
    numreader--;
    if(numreader == 0) {
        sem_post(&wrt); // If this is the last reader, it will wake up the writer.
    }
    pthread_mutex_unlock(&mutex);
}


void* writer(void* wno)
{
            sem_wait(&wrt);
    cnt = cnt*2;
    printf("Writer %d modified cnt to %d\n",(*((int *)wno)),cnt);
    sem_post(&wrt);
}
int main()
{
            pthread_t read[10],write[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt,0,1);

    int a[10] = {1,2,3,4,5,6,7,8,9,10}; //Just used for numbering the producer and consumer

    for(int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
    }

    for(int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);

    return 0;
}
```

Output :



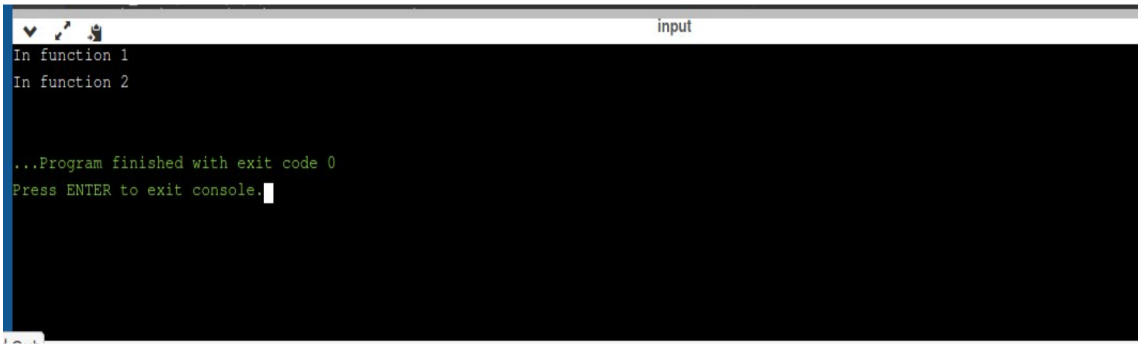
```
Reader 1: read cnt as 1
Reader 3: read cnt as 1
Writer 1 modified cnt to 2
Writer 3 modified cnt to 4
Writer 4 modified cnt to 8
Writer 2 modified cnt to 16
Reader 10: read cnt as 16
Writer 5 modified cnt to 32
Reader 9: read cnt as 32
Reader 8: read cnt as 32
Reader 6: read cnt as 32
Reader 4: read cnt as 32
Reader 2: read cnt as 32
Reader 7: read cnt as 32
Reader 5: read cnt as 32


...Program finished with exit code 0
Press ENTER to exit console.
```

3. Write a Code to access a shared resource which causes deadlock using improper use of semaphore.

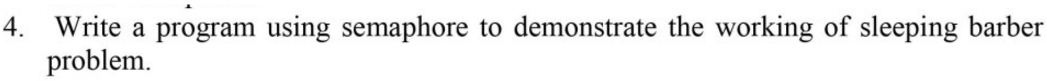
Program :

```
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>
#include<stdlib.h>
#include<unistd.h>

int shared;
sem_t sem1,sem2;

void* func1()
{
        sem_wait(&sem1);
        printf("In function 1\n");
        sem_wait(&sem2);
        sem_post(&sem2);
        sem_post(&sem1);
}

void* func2()
{
        sem_wait(&sem2);
        printf("In function 2\n");
        sem_wait(&sem1);
        sem_post(&sem1);
        sem_post(&sem2);
}

void main()
{
        pthread_t tid1,tid2;
        sem_init(&sem1,0,1);
        sem_init(&sem2,0,1);
        pthread_create(&tid1,NULL,func1,NULL);
        pthread_create(&tid2,NULL,func2,NULL);
        pthread_join(tid1,NULL);
        pthread_join(tid2,NULL);
}
```

Output :



```
In function 1
In function 2


...Program finished with exit code 0
Press ENTER to exit console.
```



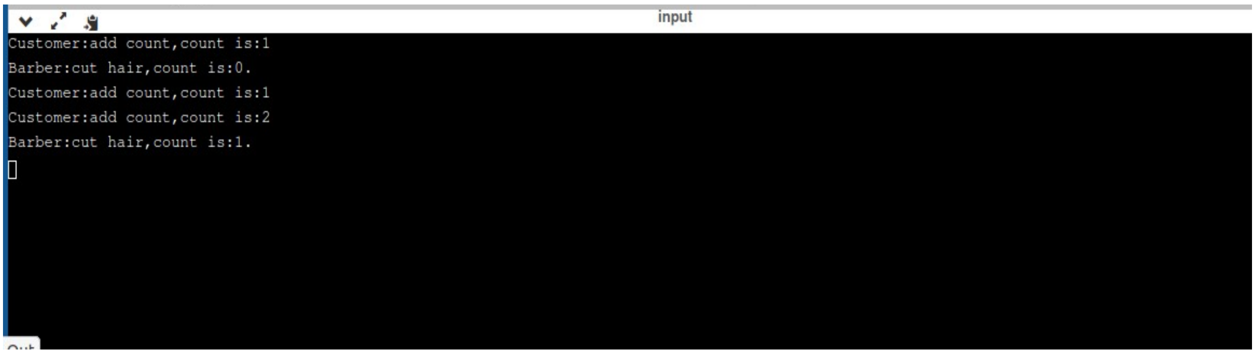
4. Write a program using semaphore to demonstrate the working of sleeping barber problem.

Program :

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<errno.h>
#include<sys/ipc.h>
#include<semaphore.h>

#define N 5

time_t end_time;/*end time*/
sem_t mutex,customers,barbers;/*Three semaphors*/
int count=0;/*The number of customers waiting for haircuts*/

void barber(void *arg);
void customer(void *arg);

int main(int argc,char *argv[])
{
        pthread_t id1,id2;
        int status=0;
        end_time=time(NULL)+20;/*Barber Shop Hours is 20s*/

        /*Semaphore initialization*/
        sem_init(&mutex,0,1);
        sem_init(&customers,0,0);
        sem_init(&barbers,0,1);

        /*Barber_thread initialization*/
        status=pthread_create(&id1,NULL,(void *)barber,NULL);
        if(status!=0)
                perror("create barbers is failure!\n");
        /*Customer_thread initialization*/
        status=pthread_create(&id2,NULL,(void *)customer,NULL);
        if(status!=0)
                perror("create customers is failure!\n");

        /*Customer_thread first blocked*/
        pthread_join(id2,NULL);
        pthread_join(id1,NULL);

        exit(0);
}

void barber(void *arg)/*Barber Process*/
{
        while(time(NULL)<end_time || count>0)
        {
                sem_wait(&customers);/*P(customers)*/
                sem_wait(&mutex);/*P(mutex)*/
                count--;
                printf("Barber:cut hair,count is:%d.\n",count);
                sem_post(&mutex);/*V(mutex)*/
                sem_post(&barbers);/*V(barbers)*/
                sleep(3);
        }
}

void customer(void *arg)/*Customers Process*/
{
        while(time(NULL)<end_time)
        {
                sem_wait(&mutex);/*P(mutex)*/
                if(count<N)
                {
                        count++;
                        printf("Customer:add count,count is:%d\n",count);
                        sem_post(&mutex);/*V(mutex)*/
                        sem_post(&customers);/*V(customers)*/
                        sem_wait(&barbers);/*P(barbers)*/
                }
                else
                        /*V(mutex)*/
                        /*If the number is full of customers,just put the mutex lock let go*/
                        sem_post(&mutex);
                sleep(1);
        }
```

}

Output :



```
Customer:add count,count is:1
Barber:cut hair,count is:0.
Customer:add count,count is:1
Customer:add count,count is:2
Barber:cut hair,count is:1.
```