SEC : A **<u>IPC-3: DEADLOCK, LOCKING, SYNCHRONIZATION</u>** **BATCH : A3**

REG: 190905513 **<u>LAB 08</u>** **NAME: MOHAMMAD DANISH EQBAL**

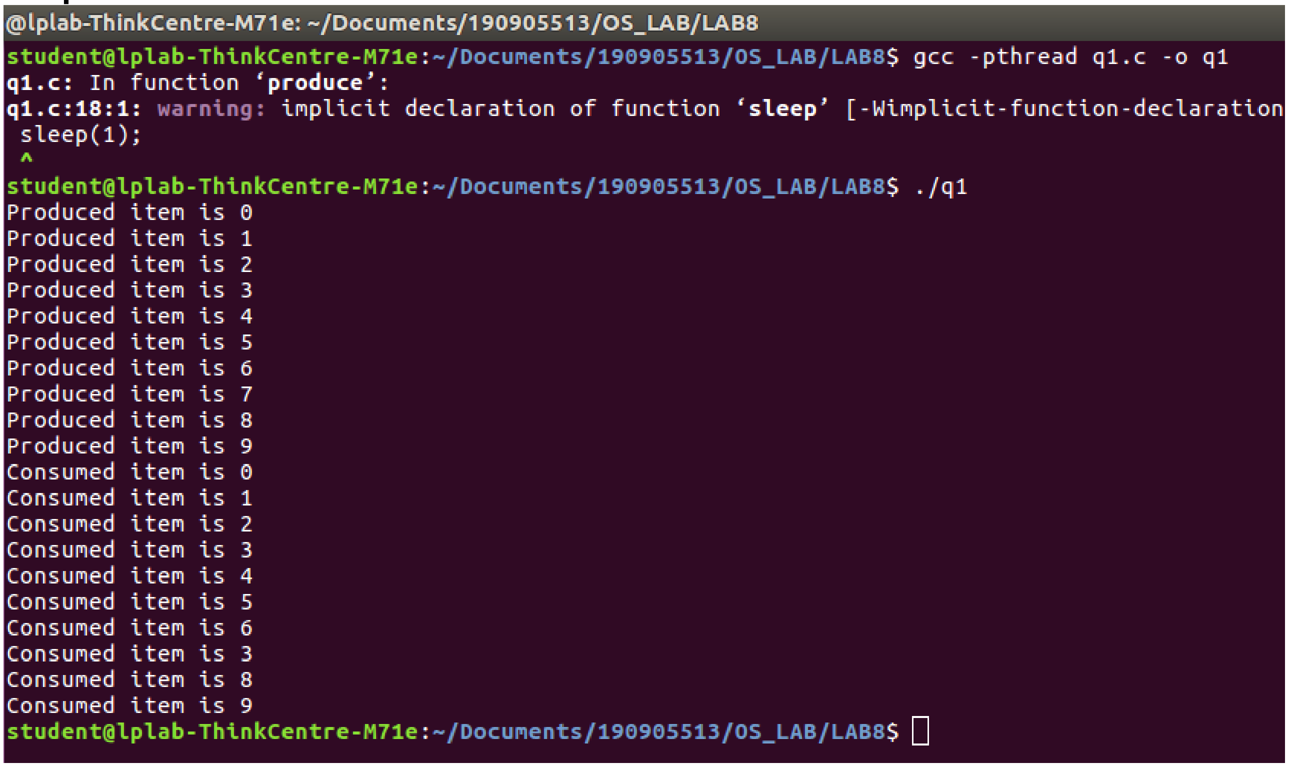
<u>LAB EXERCISES:</u>

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

Program:

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

int buf[5], f, r;
sem_t mutex, full, empty;

void* produce(void* arg)
{
for(int i=0; i<10; i++)
{
sem_wait(&empty);
sem_wait(&mutex);

printf("Produced item is %d\n", i);

buf[(++r) % 10] = i;
sleep(1);

sem_post(&mutex);
sem_post(&full);

}
}



void* consume(void* arg)
{
int item;

for(int i=0; i<10; 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()
{
pthread_t t1, t2;
sem_init(&mutex, 0, 1);
sem_init(&full, 0, 1);
sem_init(&empty, 0, 10);pthread_create(&t1, NULL, produce, NULL);
pthread_create(&t2, NULL, consume, NULL);
pthread_join(t1, NULL);
pthread_join(t2, NULL);
}
```

Output:



```
@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB8
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ gcc -pthread q1.c -o q1
q1.c: In function ‘produce’:
q1.c:18:1: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration
 sleep(1);
 ^
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ ./q1
Produced item is 0
Produced item is 1
Produced item is 2
Produced item is 3
Produced item is 4
Produced item is 5
Produced item is 6
Produced item is 7
Produced item is 8
Produced 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 3
Consumed item is 8
Consumed item is 9
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$
```

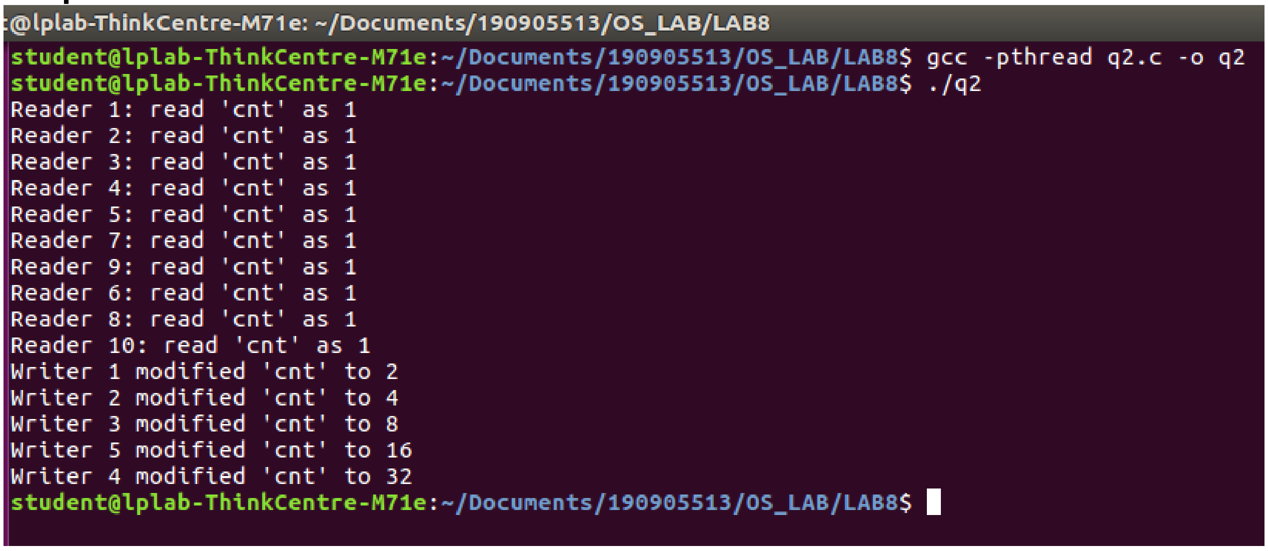
2. Write a C program for the first readers-writers problem using semaphores.

Program:

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

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

void *writer(void *wno)
{
sem_wait(&wrt);cnt *= 2;
printf("Writer %d modified 'cnt' to %d\n", (*((int *)wno)), cnt);
sem_post(&wrt);
}

void *reader(void *rno)
{
pthread_mutex_lock(&mutex);
numreader++;

if(numreader == 1)
sem_wait(&wrt); // first reader will block the writer
pthread_mutex_unlock(&mutex);

// Reading Section, no locks

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);
}

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}; //used for numbering the
producer and consumer

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



```
:@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB8
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ gcc -pthread q2.c -o q2
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ ./q2
Reader 1: read 'cnt' as 1
Reader 2: read 'cnt' as 1
Reader 3: read 'cnt' as 1
Reader 4: read 'cnt' as 1
Reader 5: read 'cnt' as 1
Reader 7: read 'cnt' as 1
Reader 9: read 'cnt' as 1
Reader 6: read 'cnt' as 1
Reader 8: read 'cnt' as 1
Reader 10: read 'cnt' as 1
Writer 1 modified 'cnt' to 2
Writer 2 modified 'cnt' to 4
Writer 3 modified 'cnt' to 8
Writer 5 modified 'cnt' to 16
Writer 4 modified 'cnt' to 32
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$
```

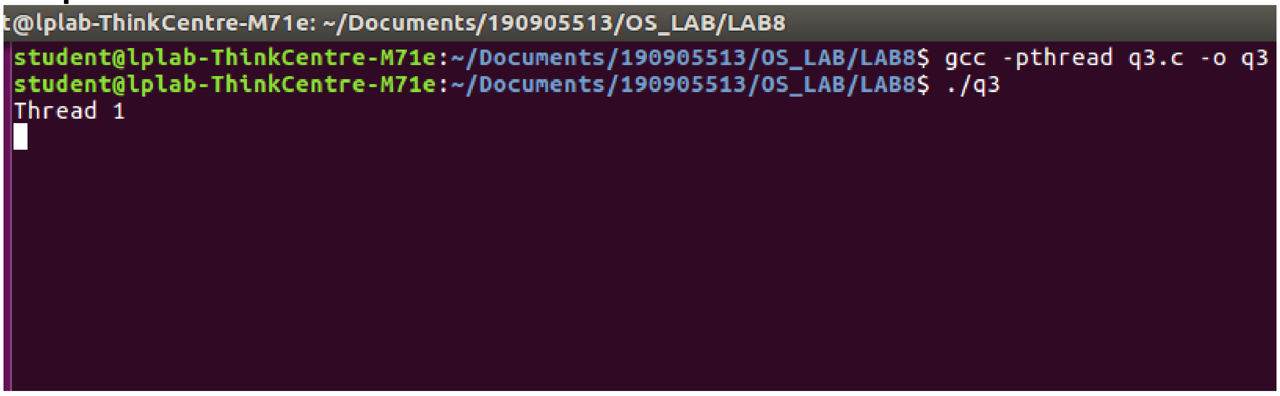
3. Write a code to access a shared resource which causes deadlock using improper use of semaphore.

Program:

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

sem_t s1,s2;
void *func1(void *p)
{
sem_wait(&s1);
sem_wait(&s2);
printf("Thread 1\n");
sem_post(&s1);
}

void *func2(void *p)
{
sem_wait(&s2);
sem_wait(&s1);
printf("Thread 2\n");
sem_post(&s2);
}

int main()
```

```
{
pthread_t threads[2];
sem_init(&s1,0,1);
sem_init(&s2,0,1);
pthread_create(&threads[0],0,func1,0);
pthread_create(&threads[1],0,func2,0);
pthread_join(threads[0],0);
pthread_join(threads[1],0);sem_destroy(&s1);
sem_destroy(&s2);
}
```

Output:



```
t@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB8
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ gcc -pthread q3.c -o q3
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ ./q3
Thread 1
```

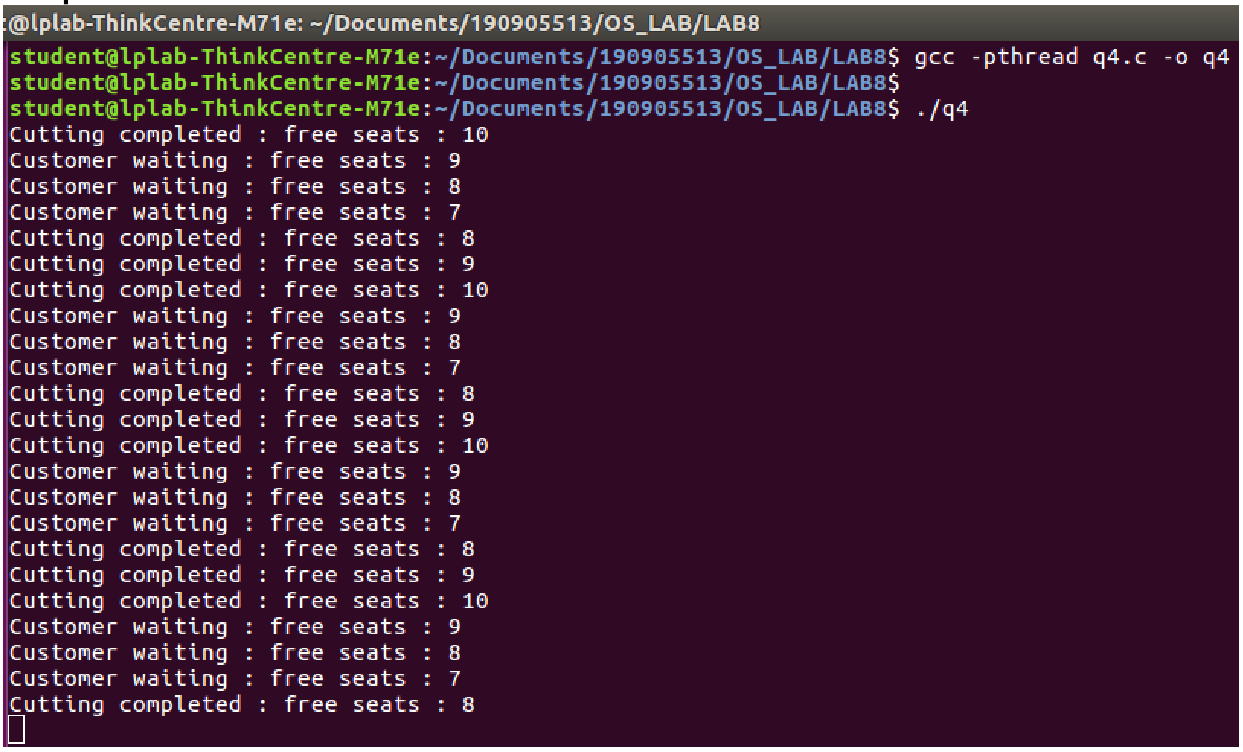
4. Write a program using semaphore to demonstrate the working of sleeping barber problem.

Program:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>
sem_t customer,barber;
pthread_mutex_t seat;
int free1 = 10;
void *br(void *args)
{
while(1)
{
sem_wait(&customer);
pthread_mutex_lock(&seat);
if(free1<10)
free1++;
sleep(2);
printf("Cutting completed : free seats : %d\n",free1);
sem_post(&barber);
pthread_mutex_unlock(&seat);
}
}
```

```
void *cr(void *args)
{
while(1)
{
pthread_mutex_lock(&seat);
if(free1 > 0)
{
free1--;
printf("Customer waiting : free seats : %d\n",free1);
sem_post(&customer);
pthread_mutex_unlock(&seat);
sem_wait(&barber);
}
else
pthread_mutex_unlock(&seat);
}
}
int main()
{
pthread_t threads[2];
sem_init(&barber,0,1);
sem_init(&customer,0,1);
pthread_mutex_init(&seat,0);
pthread_create(&threads[0],NULL,br,NULL);
pthread_create(&threads[1],NULL,cr,NULL);
pthread_join(threads[0],NULL);
pthread_join(threads[1],NULL);
sem_destroy(&barber);
sem_destroy(&customer);
pthread_mutex_destroy(&seat);
}
```

Output:



```
:@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB8
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ gcc -pthread q4.c -o q4
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB8$ ./q4
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
```