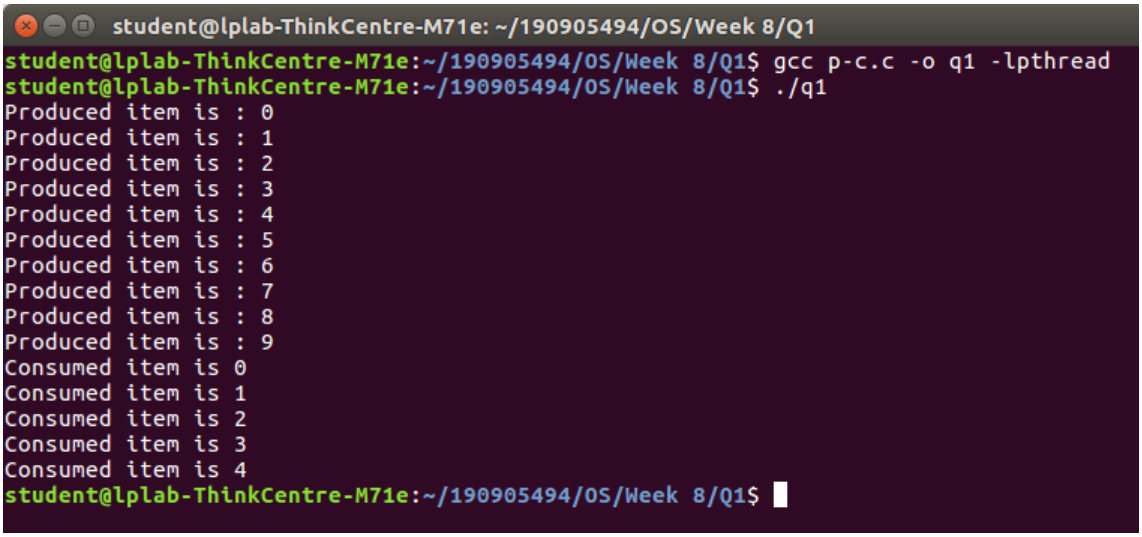
OS Lab #8

Question 1

```
/*
Producer-Consumer Code where producer can produce
at most 10 more items that consumer has consumed
*/

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

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

void* producer(void* args)
{
	for(int i=0;i<10;i++)
	{
		sem_wait(&empty);
		sem_wait(&mutex);
		printf("Produced item is : %d\n",i);
		buff[(r++)%10] = i;
		sleep(1);
		sem_post(&mutex);
		sem_post(&full);
	}
}

void* consumer(void* args)
{
	int item;
	for(int i=0;i<5;i++)
	{
		sem_wait(&full);
		sem_wait(&mutex);
		item = buff[(f++)%10];
		printf("Consumed item is %d\n",item);
		sleep(1);
		sem_post(&mutex);
		sem_post(&full);
	}
}

int main()
{
	pthread_t t1,t2;
	sem_init(&mutex,0,1);
	sem_init(&full,0,1);
	sem_init(&empty,0,10);
```

```
        pthread_create(&t1,0,producer,0);
        pthread_create(&t2,0,consumer,0);

        pthread_join(t1,0);
        pthread_join(t2,0);
}
```



Question 2

```
/*
Implementing the Reader - Writer
Problem using semaphores
*/

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

sem_t wrt;
pthread_mutex_t mutex;
int count = 1, numreader = 0;

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

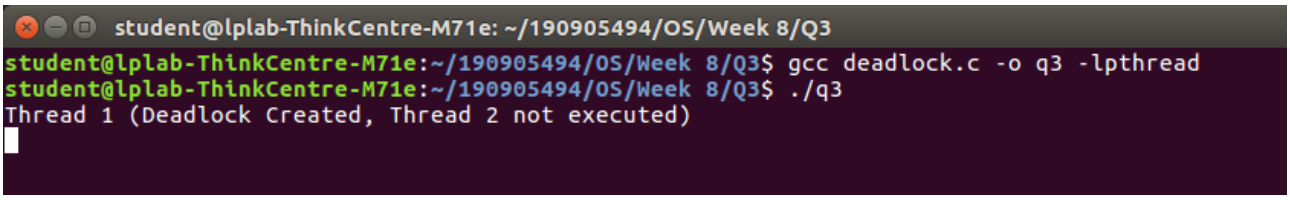
```
void* reader(void* rno)
{
	pthread_mutex_lock(&mutex);
	numreader++;

	if(numreader == 1)
		sem_wait(&wrt);

	pthread_mutex_unlock(&mutex);

	printf("Reader %d read count as %d\n",*((int*)rno),count);
	pthread_mutex_lock(&mutex);
	numreader--;

	if(numreader == 0)
		sem_post(&wrt);

	pthread_mutex_unlock(&mutex);
}

int main()
{
	pthread_t read[10],write[5];
	pthread_mutex_init(&mutex,0);
	sem_init(&wrt,0,1);

	int a[10] = {1,2,3,4,5,6,7,8,9,10};

	for(int i = 0;i<10;i++)
		pthread_create(&read[i],0,reader,&a[i]);

	for(int i = 0;i<5;i++)
		pthread_create(&write[i],0,writer,&a[i]);

	for(int i = 0;i<10;i++)
		pthread_join(read[i],0);

	for(int i = 0;i<5;i++)
		pthread_join(write[i],0);

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


}
```

```
student@lplab-ThinkCentre-M71e: ~/190905494/OS/Week 8/Q2
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 8/Q2$ gcc r-w.c -o q2 -lpthread
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 8/Q2$ ./q2
Reader 2 read count as 1
Reader 10 read count as 1
Reader 3 read count as 1
Reader 4 read count as 1
Reader 5 read count as 1
Reader 6 read count as 1
Reader 7 read count as 1
Reader 8 read count as 1
Reader 9 read count as 1
Reader 1 read count as 1
Writer 3 modified count to 2
Writer 4 modified count to 4
Writer 5 modified count to 8
Writer 1 modified count to 16
Writer 2 modified count to 32
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 8/Q2$
```

Question 3

```
/*
Creating Deadlock while accessing shared Resorce,
by the improper use of semaphores
*/

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

sem_t s1,s2;

void* func1(void* p)
{
	sem_wait(&s1);
	sem_wait(&s2);
	printf("Thread 1 (Deadlock Created, Thread 2 not executed)\n");
	sem_post(&s1);
}

void* func2(void* p)
{
	sem_wait(&s2);
	sem_wait(&s1);
	printf("Thread 2 (Deadlock Created, Thread 1 not executed)\n");
	sem_post(&s2);
}

int main()
{
	pthread_t t1,t2;
	sem_init(&s1,0,1);
	sem_init(&s2,0,1);
```

```
	pthread_create(&t1,0,func1,0);
	pthread_create(&t2,0,func2,0);

	pthread_join(t1,0);
	pthread_join(t2,0);

	sem_destroy(&s1);
	sem_destroy(&s2);
}
```



Question 4

```
/*
Using semaphores to demonstrate the
working of the sleeping barber problem
*/


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

sem_t customer, barber;
pthread_mutex_t seat;
int free0 = 10;

void* br(void* args)
{
	while(1)
	{
		sem_wait(&customer);
		pthread_mutex_lock(&seat);

		if(free0 < 10)
			free0++;

		sleep(1);
		printf("Cutting completed : Free seats : %d\n",free0);
		sem_post(&barber);
		pthread_mutex_unlock(&seat);
	}
}
```

```
void* cr(void* args)
{
	while(1)
	{
		pthread_mutex_lock(&seat);

		if(free0 > 0)
		{
			free0--;
			printf("Customer waiting : Free seats : %d\n",free0);
			sem_post(&customer);
			pthread_mutex_unlock(&seat);
			sem_wait(&barber);
		}
		else
			pthread_mutex_unlock(&seat);
	}
}


int main()
{
	pthread_t t1,t2;
	sem_init(&barber,0,1);
	sem_init(&customer,0,1);

	pthread_mutex_init(&seat,0);
	pthread_create(&t1,0,br,0);
	pthread_create(&t2,0,cr,0);

	pthread_join(t1,0);
	pthread_join(t2,0);

	sem_destroy(&barber);
	sem_destroy(&customer);
	pthread_mutex_destroy(&seat);
}
```

```
student@lplab-ThinkCentre-M71e: ~/190905494/OS/Week 8/Q4
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 8/Q4$ gcc barber.c -o q4 -lpthread
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 8/Q4$ ./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
```