Ayush Goyal
190905522 CSE D Roll 62

Operating Systems Week 8: Lab 7: IPC-3: Deadlock, Locking, Synchronization

Lab Excercises:

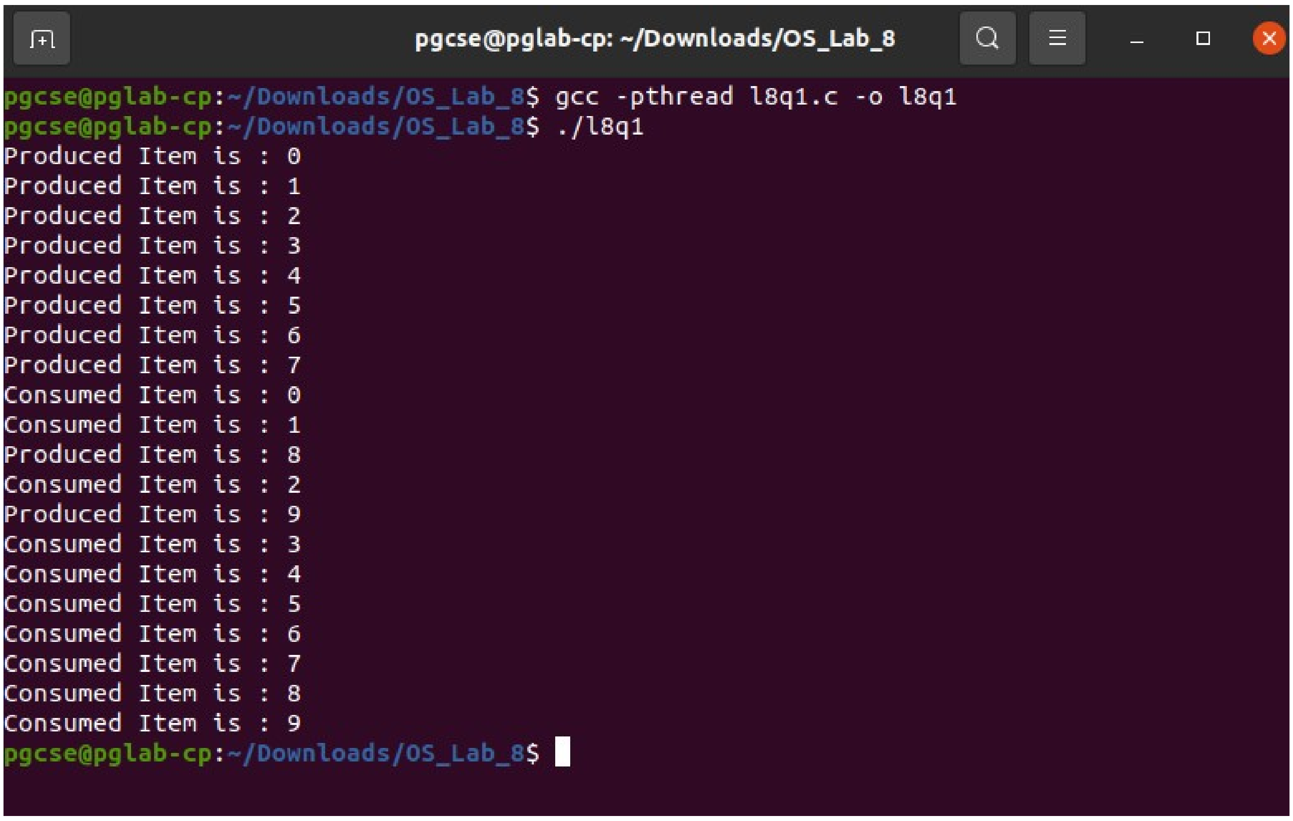
1. Modify the above Producer-Consumer program so that, a producer can produce at the most 10 items more than what the consumer has consumed.

Code:

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

int buf[5], f, r;
sem_t mutex_sem, end, start;

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

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

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

		sem_post(&mutex_sem);
		sem_post(&end);
	}
}

void *consume(void *arg){
	int item;
	for(int i=0;i<10;i++){
		sem_wait(&end);
		sem_wait(&mutex_sem);

		item = buf[(++f)%10];
		printf("Consumed Item is : %d\n", item);
		sleep(1);

		sem_post(&mutex_sem);
		sem_post(&start);
	}
}
```

```
int main(){
        pthread_t t1, t2;
        sem_init(&mutex_sem, 0, 1);
        sem_init(&end, 0, 1);
        sem_init(&start, 0, 10);
        pthread_create(&t1, NULL, &produce, NULL);
        pthread_create(&t2, NULL, &consume, NULL);
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
        return 0;
}
```

Output:



```
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ gcc -pthread l8q1.c -o l8q1
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ ./l8q1
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
Consumed Item is : 0
Consumed Item is : 1
Produced Item is : 8
Consumed Item is : 2
Produced Item is : 9
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
pgcse@pglab-cp:~/Downloads/OS_Lab_8$
```

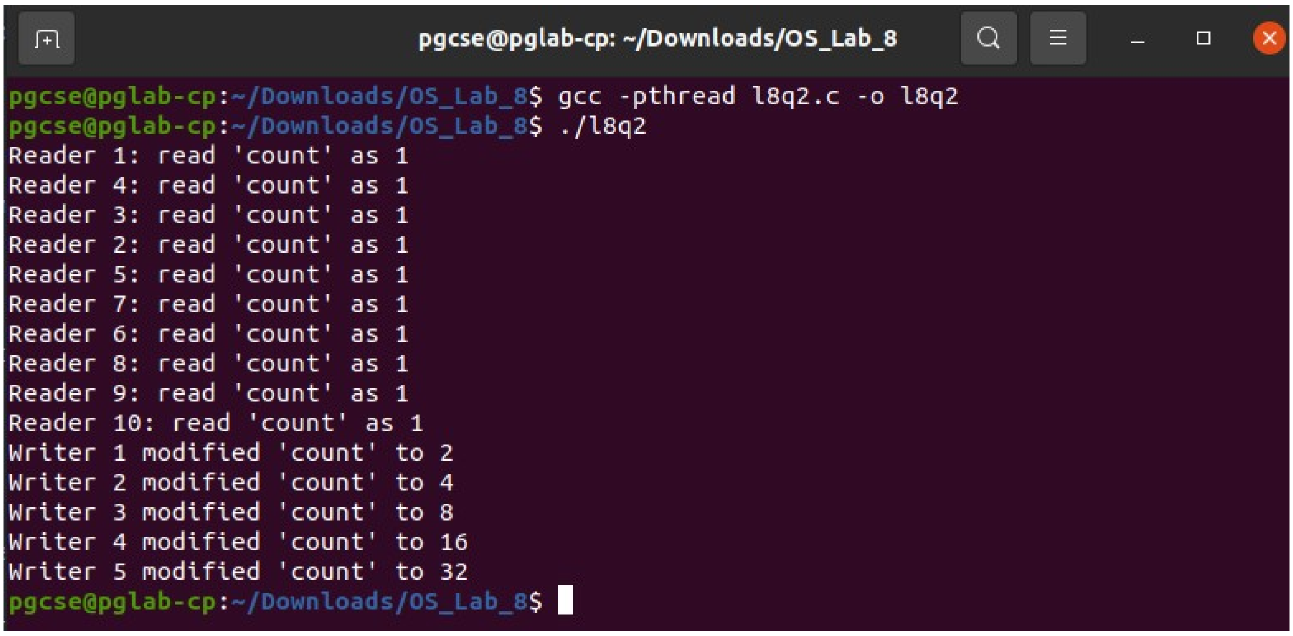
2. Write a C program for the first readers-writers problem using semaphores.

Code:

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

sem_t wrt;
pthread_mutex_t mutex_sem;
int count = 1;
int 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_sem);
        numreader++;

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

        //Reading Section, hence no locks

        printf("Reader %d: read 'count' as %d\n", *((int*)rno), count);

        //Reader acquire the lock before modifying numreader

        pthread_mutex_lock(&mutex_sem);
        numreader--;

        if(numreader == 0)
                sem_post(&wrt); //If this is the last reader, it will wake up the writer
        pthread_mutex_unlock(&mutex_sem);
}

int main(){
        pthread_t read[10], write[5];
        pthread_mutex_init(&mutex_sem, NULL);
        sem_init(&wrt, 0, 1);
        int a[10] = {1,2,3,4,5,6,7,8,9,10}; //used for numbering the producers and consumers

        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);
        sem_destroy(&wrt);
        return 0;
}
```

Output:



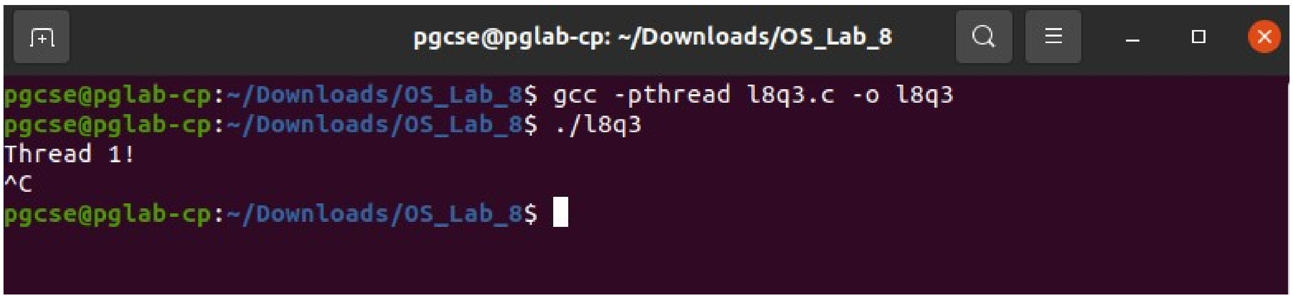
```
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ gcc -pthread l8q2.c -o l8q2
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ ./l8q2
Reader 1: read 'count' as 1
Reader 4: read 'count' as 1
Reader 3: read 'count' as 1
Reader 2: read 'count' as 1
Reader 5: read 'count' as 1
Reader 7: read 'count' as 1
Reader 6: read 'count' as 1
Reader 8: read 'count' as 1
Reader 9: read 'count' as 1
Reader 10: read 'count' as 1
Writer 1 modified 'count' to 2
Writer 2 modified 'count' to 4
Writer 3 modified 'count' to 8
Writer 4 modified 'count' to 16
Writer 5 modified 'count' to 32
pgcse@pglab-cp:~/Downloads/OS_Lab_8$
```

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

Code:

```
#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!\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:



```
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ gcc -pthread l8q3.c -o l8q3
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ ./l8q3
Thread 1!
^C
pgcse@pglab-cp:~/Downloads/OS_Lab_8$
```

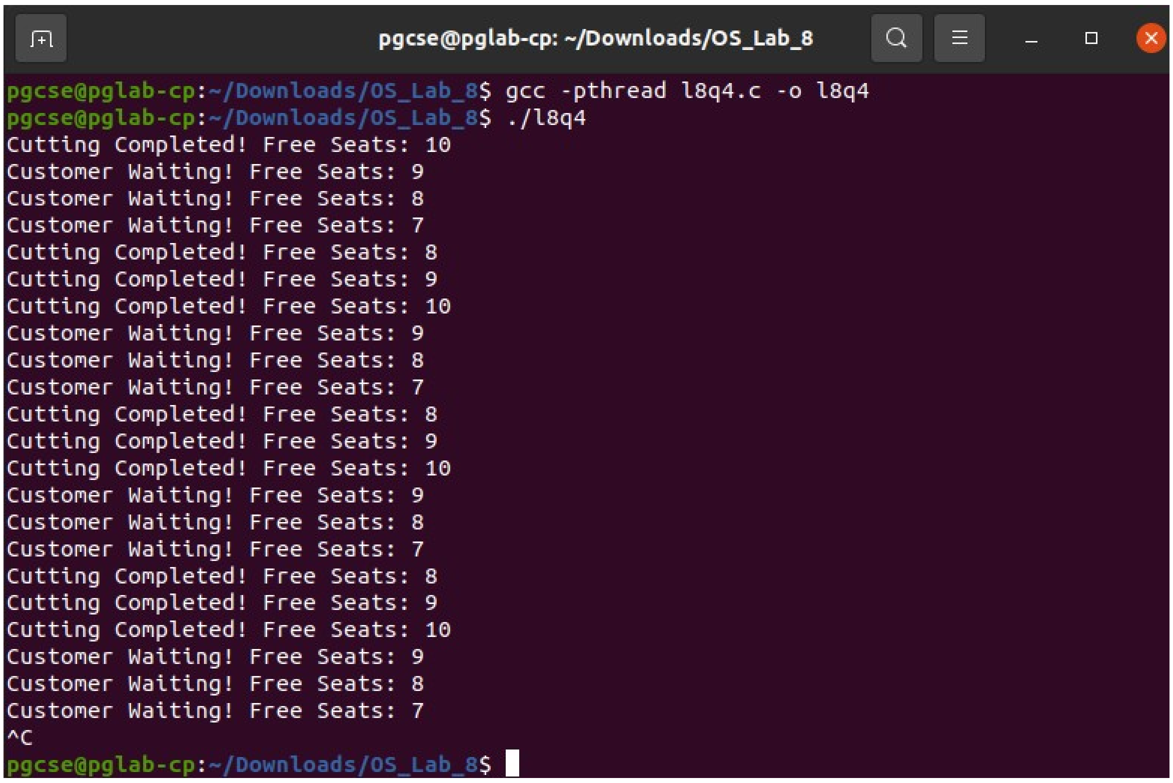
4. Write a program using semaphore to demonstrate the working of sleeping barber problem.

Code:

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

sem_t customer, barber;
pthread_mutex_t seat;
int free1 = 10;

void *barb(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 *cust(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, &barb, NULL);
	pthread_create(&threads[1], NULL, &cust, NULL);
	pthread_join(threads[0], NULL);
	pthread_join(threads[1], NULL);
	sem_destroy(&barber);
	sem_destroy(&customer);
	pthread_mutex_destroy(&seat);
}
```

Output:



```
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ gcc -pthread l8q4.c -o l8q4
pgcse@pglab-cp:~/Downloads/OS_Lab_8$ ./l8q4
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
^C
pgcse@pglab-cp:~/Downloads/OS_Lab_8$
```

THE END