OS Lab
Rhea Adhikari
190905156
Batch D1
Lab8

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.

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

int buffer[5], f, r;

sem_t mutex, full, empty;

void *produce(void *arg)
{
   for (int i = 0; i < 15; i++)
   {
      sem_wait(&empty);
      sem_wait(&mutex);
      printf("Item produced is: '%d'\n", i);
      buffer[(++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 = buffer[(++f) % 10];
      printf("Item consumed is: '%d'\n", item);
      sleep(1);
      sem_post(&mutex);
      sem_post(&empty);
   }
}
int main()
{
   pthread_t t1;
   pthread_t 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);
}
```

```
Student@project-lab:~/Documents/190905156/OS/Lab8$ gcc first.c -o first -pthread
first.c: In function ‘produce’:
first.c:17:9: warning: implicit declaration of function ‘sleep’; did you mean ‘select’? [-Wimplicit-function-declaration]
         sleep(1);
         ^~~~~
         select
Student@project-lab:~/Documents/190905156/OS/Lab8$ ./first
Item produced is: '0'
Item produced is: '1'
Item produced is: '2'
Item produced is: '3'
Item produced is: '4'
Item produced is: '5'
Item produced is: '6'
Item produced is: '7'
Item produced is: '8'
Item produced is: '9'
Item consumed is: '0'
Item consumed is: '1'
Item consumed is: '2'
Item consumed is: '3'
Item consumed is: '4'
Item consumed is: '5'
Item consumed is: '6'
Item consumed is: '7'
Item consumed is: '8'
Item consumed is: '9'
Item produced is: '10'
Item produced is: '11'
Item produced is: '12'
Item produced is: '13'
Item produced is: '14'
Student@project-lab:~/Documents/190905156/OS/Lab8$
```

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

```
#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);
        pthread_mutex_unlock(&mutex);
        printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
        pthread_mutex_lock(&mutex);
        numReader--;
        if(numReader == 0)
        sem_post(&wrt);
        pthread_mutex_unlock(&mutex);
```

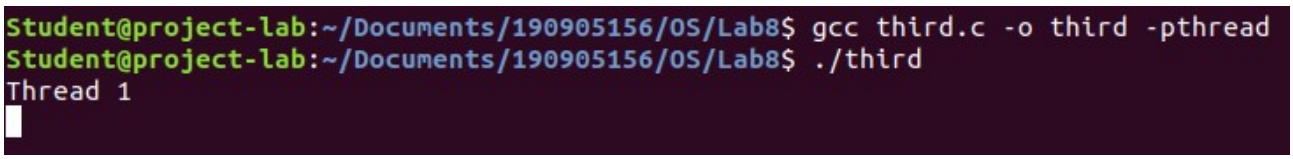
```
}

int main()
{
        pthread_t read[10],write[5];
        pthread_mutex_init(&mutex, NULL);
        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], 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;
}
```

```
Student@project-lab:~/Documents/190905156/OS/Lab8$ gcc second.c -o second -pthread
Student@project-lab:~/Documents/190905156/OS/Lab8$ ./second
Reader 2: read cnt as 1
Reader 4: read cnt as 1
Reader 5: read cnt as 1
Reader 3: read cnt as 1
Reader 6: read cnt as 1
Reader 1: read cnt as 1
Reader 7: read cnt as 1
Reader 8: read cnt as 1
Reader 9: read cnt as 1
Reader 10: read cnt as 1
Writer 2 modified cnt to 2
Writer 3 modified cnt to 4
Writer 1 modified cnt to 8
Writer 4 modified cnt to 16
Writer 5 modified cnt to 32
Student@project-lab:~/Documents/190905156/OS/Lab8$
```

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

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

sem_t s1,s2;

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

void *functionTwo(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,functionOne,0);
        pthread_create(&threads[1],0,functionTwo,0);
        pthread_join(threads[0],0);
        pthread_join(threads[1],0);sem_destroy(&s1);
        sem_destroy(&s2);
}
```



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

```
#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);
}
```

```
Student@project-lab:~/Documents/190905156/OS/Lab8$ gcc fourth.c -o fourth -pthread
Student@project-lab:~/Documents/190905156/OS/Lab8$ ./fourth
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
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
```