190905514 **MOHAMMAD TOFIK** **BATCH : C3**
ROLLNO : 62 **SECTION : C** **SEM : 5th**

:- OS-LAB-8 :-

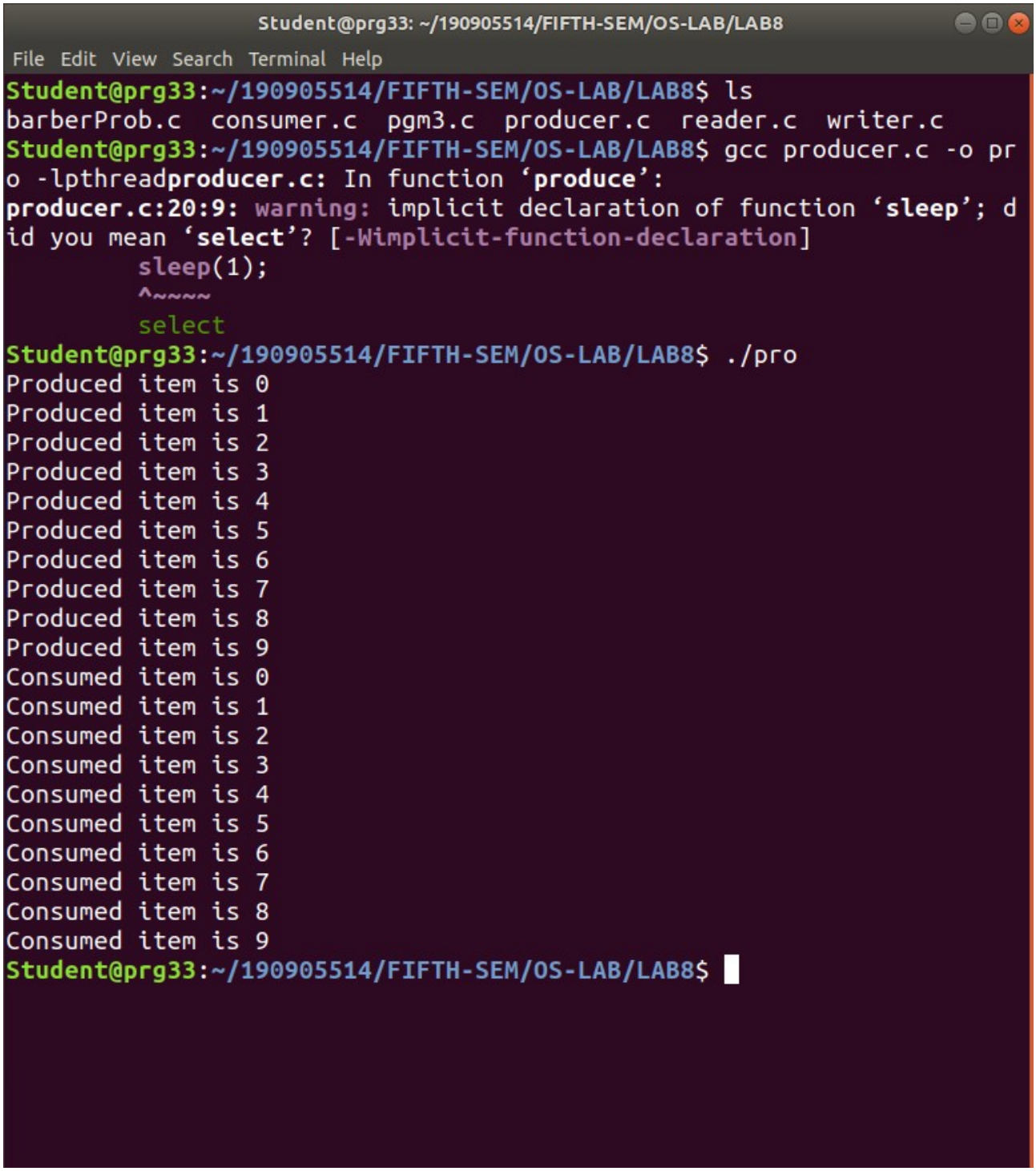
EXCERCISE :

pgm1.c

```
/*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>
#include <stdlib.h>
#include <string.h>

int buf[1024], 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 :

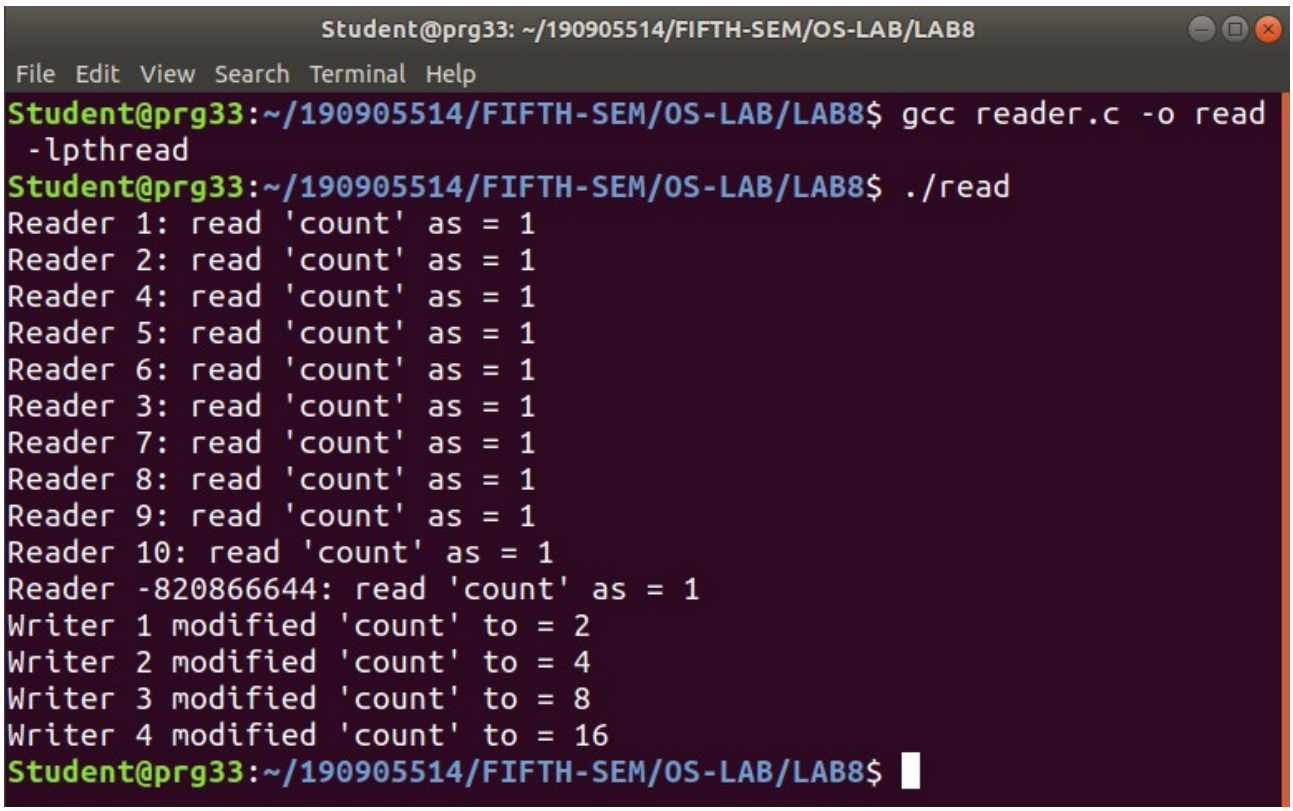

Student@prg33: ~/190905514/FIFTH-SEM/OS-LAB/LAB8
File Edit View Search Terminal Help
Student@prg33:~/190905514/FIFTH-SEM/OS-LAB/LAB8$ ls
barberProb.c consumer.c pgm3.c producer.c reader.c writer.c
Student@prg33:~/190905514/FIFTH-SEM/OS-LAB/LAB8$ gcc producer.c -o pr
o -lpthreadproducer.c: In function 'produce':
producer.c:20:9: warning: implicit declaration of function 'sleep'; d
id you mean 'select'? [-Wimplicit-function-declaration]
sleep(1);
^~~~~
select
Student@prg33:~/190905514/FIFTH-SEM/OS-LAB/LAB8$ ./pro
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 7
Consumed item is 8
Consumed item is 9
Student@prg33:~/190905514/FIFTH-SEM/OS-LAB/LAB8$

pgm2.c

```c
/*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 'count' 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 'count' 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 < 11; i++)
pthread_create(&read[i], NULL, reader, &a[i]);
for (int i = 0; i < 4; i++)
pthread_create(&write[i], NULL, writer, &a[i]);
for (int i = 0; i < 11; i++)
pthread_join(read[i], NULL);
for (int i = 0; i < 4; i++)
pthread_join(write[i], NULL);

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

OUTPUT :



pgm3.c

```c
/*3.) Write a Code to access a shared resource which causes deadlock
using improper use of semaphore.*/
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

sem_t semaphore;

void *func1(void *param)
{
printf("Thread 1 \n");
sem_post(&semaphore);
}

void *func2(void *param)
{
printf("Thread 2 \n");
```

```
sem_post(&semaphore);
}

int main()
{
pthread_t threads[2];
sem_init(&semaphore, 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(&semaphore);
}
```

OUTPUT :

pgm4.c

```c
/*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);
}
```

OUTPUT :

```
Student@prg33: ~/190905514/FIFTH-SEM/OS-LAB/LAB8
File Edit View Search Terminal Help
Student@prg33:~/190905514/FIFTH-SEM/OS-LAB/LAB8$ gcc pgm4.c -o pgm4 -
lpthread
Student@prg33:~/190905514/FIFTH-SEM/OS-LAB/LAB8$ ./pgm4
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
```