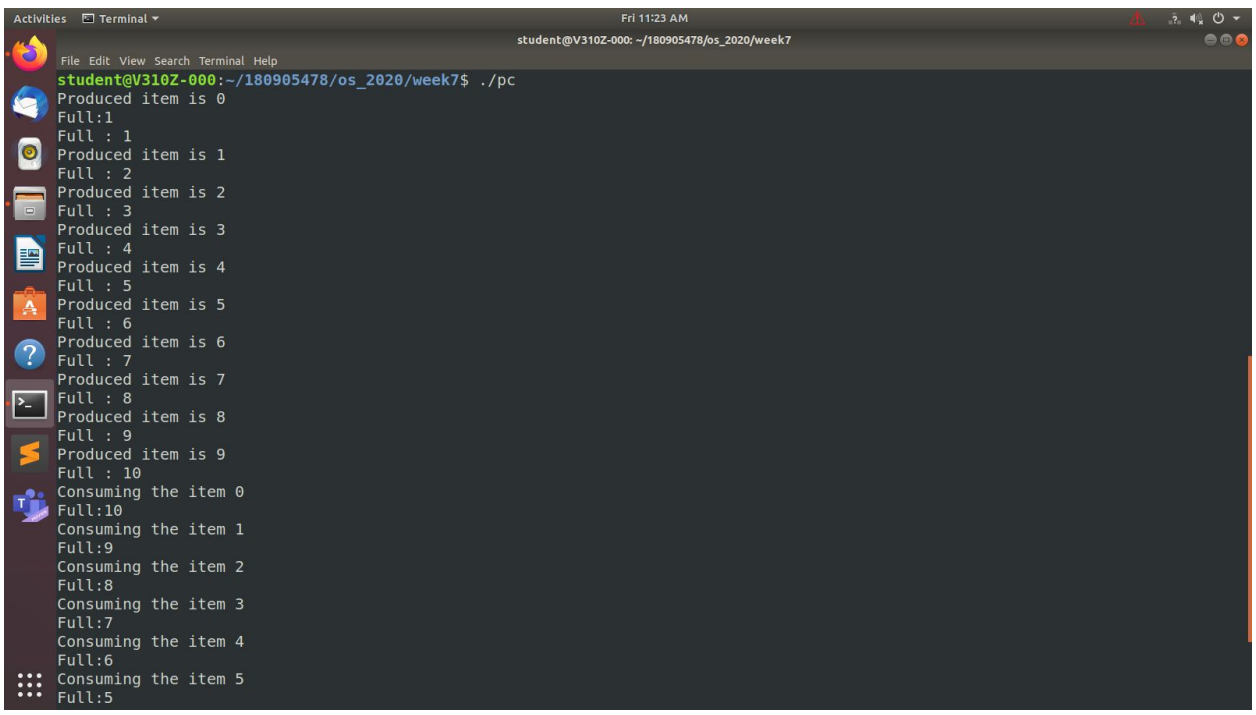
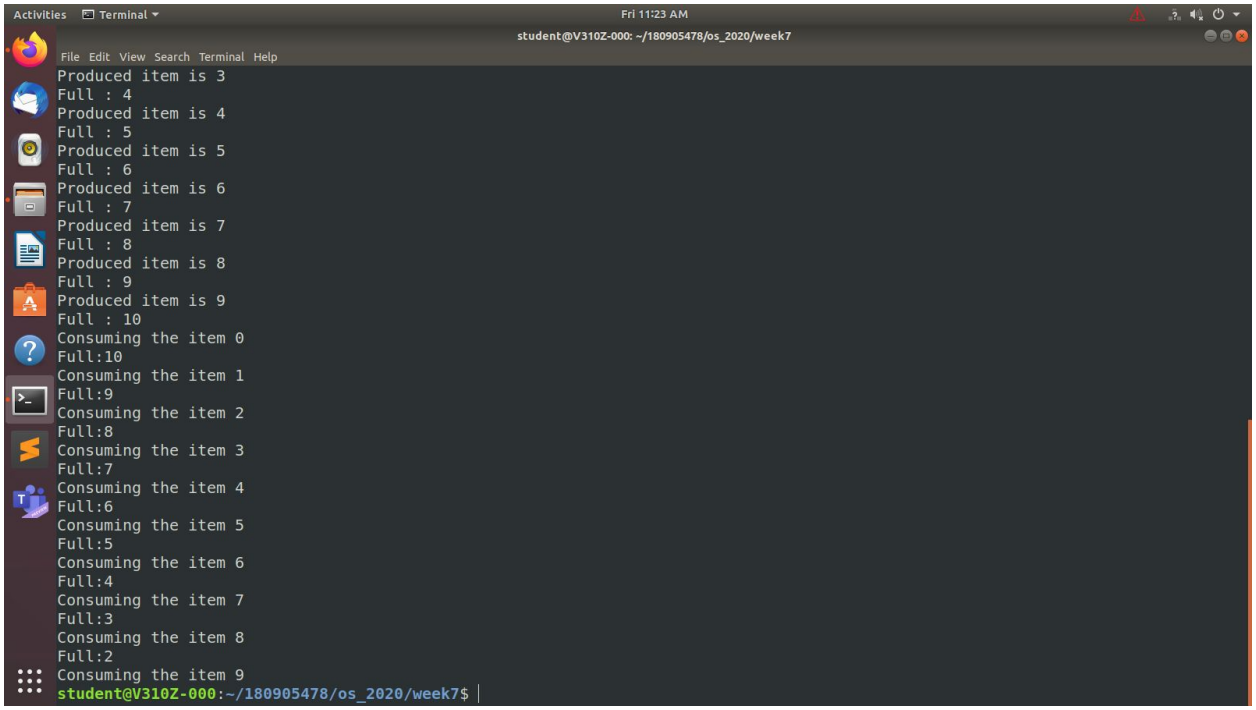
OS LAB 7
REG: 180905478
NAME :SAGNIK CHATTERJEE
ROLL NO: 61
SEC : B

Q1.

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


int buf[10], f = 0,r = 0, item, val;
sem_t mutex, full, empty;

void *producer(void *arg){
        int i;
        for(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);
        sem_getvalue(&full, &val);
        printf("Full : %d \n", val);
        }
}

void *consumer(void *arg){
        int i;
        for(i = 0; i < 10; i++){
        sem_getvalue(&full, &val);
        printf("Full:%d \n", val);
        sem_wait(&full);
        sem_wait(&mutex);
        item = buf[(++f) % 10];
        printf("Consuming the item %d\n", item);
        sleep(1);
        sem_post(&mutex);
        sem_post(&empty);
```

```
        }
}

int main(int argc, char const *argv[]){
        pthread_t tid1, tid2;
        sem_init(&mutex, 0, 1);
        sem_init(&full, 0, 1);
        sem_init(&empty, 0, 10);
        pthread_create(&tid1, NULL, producer, NULL);
        pthread_create(&tid2, NULL, consumer, NULL);
        pthread_join(tid1, NULL);
        pthread_join(tid2, NULL);
        return 0;
}
```

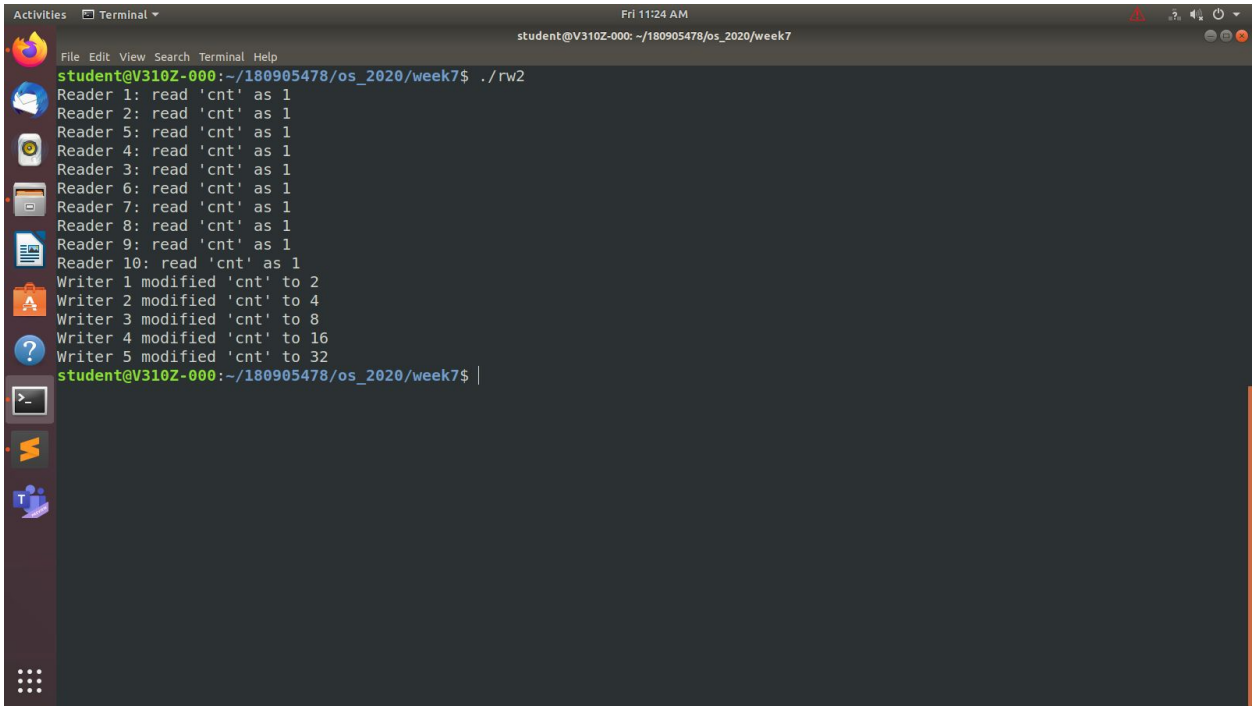




Q2

```
#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;

}
```

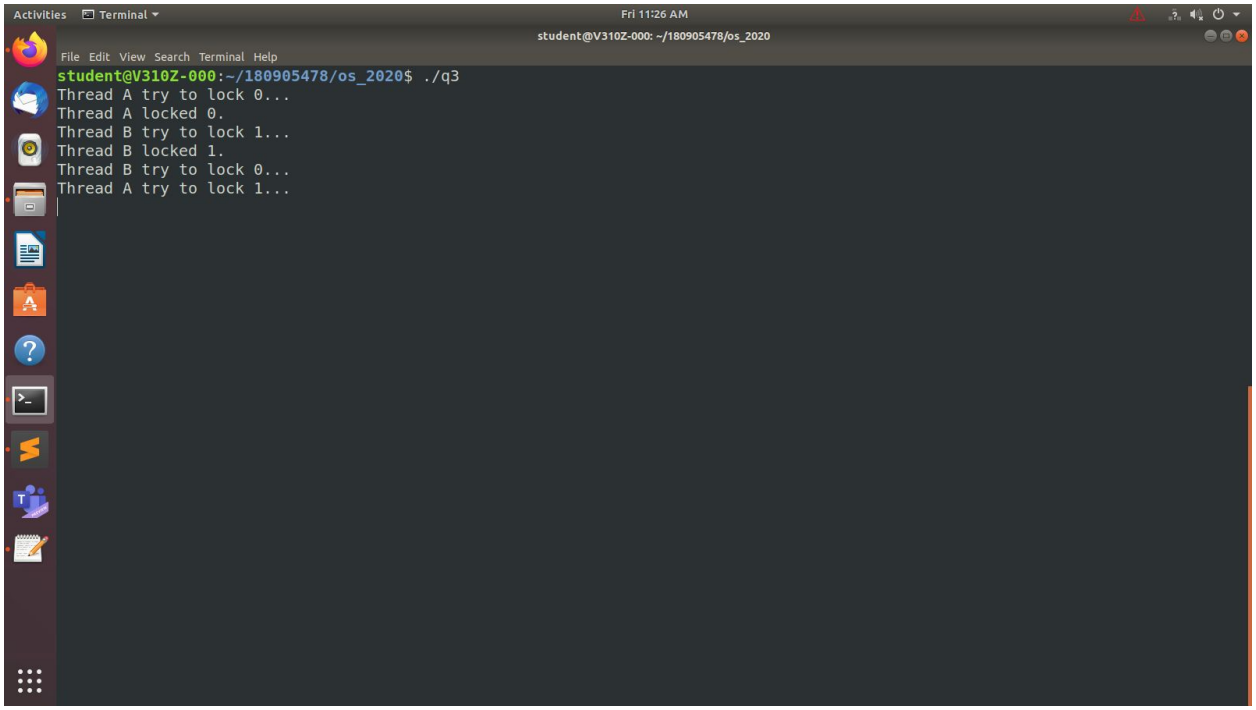


Q3

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/sem.h>

#define PERMS 0660

int semId;

int initSem(int semId, int semNum, int initValue) {
    return semctl(semId, semNum, SETVAL, initValue);
}

int P(int semId, int semNum) {
    struct sembuf operationList[1];
    operationList[0].sem_num = semNum;
    operationList[0].sem_op = -1;
    operationList[0].sem_flg = 0;
    return semop(semId, operationList, 1);
}

int V(int semId, int semNum) {
```

```
    struct sembuf operationList[1];
    operationList[0].sem_num = semNum;
    operationList[0].sem_op = 1;
    operationList[0].sem_flg = 0;
    return semop(semId, operationList, 1);
}

void* funcA(void* nothing) {
    printf("Thread A try to lock 0...\n");
    P(semId, 0);
    printf("Thread A locked 0.\n");
    usleep(50*1000);
    printf("Thread A try to lock 1...\n");
    P(semId, 1);
    printf("Thread A locked 1.\n");
    V(semId, 0);
    V(semId, 1);
return NULL;
}
void* funcB(void* nothing) {
printf("Thread B try to lock 1...\n");
P(semId, 1);
printf("Thread B locked 1.\n");
usleep(5*1000);
printf("Thread B try to lock 0...\n");
P(semId, 0);
printf("Thread B locked 0.\n");
V(semId, 0);
V(semId, 1);
return NULL;
}
int main(int argc, char* argv[]) {
int i;
semId = semget(ftok(argv[0], 'A'), 2, IPC_CREAT | PERMS);
initSem(semId, 0, 1);
initSem(semId, 1, 1);
pthread_t thread[2];
pthread_create(&thread[0], NULL, funcA, NULL);
pthread_create(&thread[1], NULL, funcB, NULL);
for (i = 0 ; i < 2 ; i++) {
pthread_join(thread[i], NULL);
}
printf("This is not printed in case of deadlock\n");
semctl(semId, 0, IPC_RMID, 0);
```

```
semctl(semId, 1, IPC_RMID, 0);
return 0;
}
```



Q4

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

#define MAX 20

void *client(void *param);
void *barber(void *param);

sem_t chairs_mutex;
sem_t sem_client;
sem_t sem_barber;
int num_chairs;
int clientWait;
int main(int argc, char *argv[]) {
    pthread_t barberid;
    pthread_t clientids[MAX];
```

```
 printf("Main thread beginning\n");
        int runTime,clients,i;
  if (argc != 5){
        printf("Please enter 4 arguments: <Program run time> <Number of clients>\n");
        printf("<Number of chairs> <Client wait time>\n");
        exit(0);
  }
  runTime = atoi(argv[1]);
  clients = atoi(argv[2]);
  num_chairs = atoi(argv[3]);
  clientWait = atoi(argv[4]);

  sem_init(&chairs_mutex,0,1);
  sem_init(&sem_client,0,0);
  sem_init(&sem_barber,0,0);

  pthread_create(&barberid, NULL, barber, NULL);
  printf("Creating barber thread with id %lu\n",barberid);

  for (i = 0; i < clients; i++){
        pthread_create(&clientids[i], NULL, client, NULL);
        printf("Creating client thread with id %lu\n",clientids[i]);
  }

  printf("Main thread sleeping for %d seconds\n", runTime);
  sleep(runTime);

  printf("Main thread exiting\n");
  exit(0);
}

void *barber(void *param) {
  int worktime;

  while(1) {
        sem_wait(&sem_client);
        sem_wait(&chairs_mutex);
    num_chairs += 1;
    printf("Barber: Taking a client. Number of chairs available = %d\n",num_chairs);
    sem_post(&sem_barber);
    sem_post(&chairs_mutex);
    worktime = (rand() % 4) + 1;
    printf("Barber: Cutting hair for %d seconds\n", worktime);
```

```
    sleep(worktime);
        }
}

void *client(void *param) {
  int waittime;

  while(1) {

        sem_wait(&chairs_mutex);

        if(num_chairs <= 0){

                printf("Client: Thread %u leaving with no haircut\n", (unsigned int)pthread_self());
                sem_post(&chairs_mutex);
    }

        else{

                num_chairs -= 1;
                printf("Client: Thread %u Sitting to wait. Number of chairs left = %d\n",(unsigned
int)pthread_self(),num_chairs);

                sem_post(&sem_client);

                sem_post(&chairs_mutex);

                sem_wait(&sem_barber);

                printf("Client: Thread %u getting a haircut\n",(unsigned int)pthread_self());
    }

        waittime = (rand() % clientWait) + 1;

        printf("Client: Thread %u waiting %d seconds before attempting next
haircut\n",(unsigned int)pthread_self(),waittime);
    sleep(waittime);
        }
}
```

```
student@V310Z-000:~/180905478/os_2020/week7$ ./sb 3 4 2 1
Main thread beginning
Creating barber thread with id 139747147835136
Creating client thread with id 139747139442432
Client: Thread 1788532480 Sitting to wait. Number of chairs left = 1
Barber: Taking a client. Number of chairs available = 2
Barber: Cutting hair for 4 seconds
Client: Thread 1780139776 Sitting to wait. Number of chairs left = 1
Client: Thread 1780139776 getting a haircut
Client: Thread 1780139776 waiting 1 seconds before attempting next haircut
Creating client thread with id 139747131049728
Creating client thread with id 139747122657024
Client: Thread 1771747072 Sitting to wait. Number of chairs left = 0
Client: Thread 1763354368 leaving with no haircut
Client: Thread 1763354368 waiting 1 seconds before attempting next haircut
Creating client thread with id 139747114264320
Main thread sleeping for 3 seconds
Client: Thread 1780139776 leaving with no haircut
Client: Thread 1780139776 waiting 1 seconds before attempting next haircut
Client: Thread 1763354368 leaving with no haircut
Client: Thread 1763354368 waiting 1 seconds before attempting next haircut
Client: Thread 1780139776 leaving with no haircut
Client: Thread 1780139776 waiting 1 seconds before attempting next haircut
Client: Thread 1763354368 leaving with no haircut
Client: Thread 1763354368 waiting 1 seconds before attempting next haircut
Main thread exiting
student@V310Z-000:~/180905478/os_2020/week7$
```