Registration Number:19BCE2119
Name: Gaurav Kumar Singh
Course: CSE2005 Operating Systems
Digital Assignment 3

(a) Implement the solution for reader – writer's problem.

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

sem_t mutex,writeblock;
int data = 0,rcount = 0;

void *reader(void *arg)
{
  int f;
  f = ((int)arg);
  sem_wait(&mutex);
  rcount = rcount + 1;
  if(rcount==1)
   sem_wait(&writeblock);
  sem_post(&mutex);
  printf("Data read by the reader%d is %d\n",f,data);
  sleep(1);
  sem_wait(&mutex);
  rcount = rcount - 1;
  if(rcount==0)
   sem_post(&writeblock);
  sem_post(&mutex);
}

void *writer(void *arg)
{
  int f;
  f = ((int) arg);
  sem_wait(&writeblock);
  data++;
  printf("Data writen by the writer%d is %d\n",f,data);
  sleep(1);
  sem_post(&writeblock);
}

int main()
{
  int i,b;
  pthread_t rtid[5],wtid[5];
  sem_init(&mutex,0,1);
  sem_init(&writeblock,0,1);
  for(i=0;i<=2;i++)
  {
    pthread_create(&wtid[i],NULL,writer,(void *)i);
    pthread_create(&rtid[i],NULL,reader,(void *)i);
```

```
  }
  for(i=0;i<=2;i++)
  {
    pthread_join(wtid[i],NULL);
    pthread_join(rtid[i],NULL);
  }
  return 0;
}
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA3
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ gcc 1.c -o 1 -lpthread
1.c: In function ‘reader’:
1.c:11:7: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
   11 |  f = ((int)arg);
      |       ^
1.c:18:2: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declaration]
   18 |  sleep(1);
      |  ^~~~~
1.c: In function ‘writer’:
1.c:29:7: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
   29 |  f = ((int) arg);
      |       ^
1.c: In function ‘main’:
1.c:45:39: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   45 |   pthread_create(&wtid[i],NULL,writer,(void *)i);
      |                                       ^
1.c:46:39: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   46 |   pthread_create(&rtid[i],NULL,reader,(void *)i);
      |                                       ^
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ ./1
Data writen by the writer0 is 1
Data read by the reader0 is 1
Data read by the reader1 is 1
Data read by the reader2 is 1
Data writen by the writer1 is 2
Data writen by the writer2 is 3
gaurav1020@DESKTOP-R0RPIEK:~/DA3$
```

(b) Implement the solution for dining philosopher’s problem.

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

sem_t room;
sem_t chopstick[5];

void * philosopher(void *);
void eat(int);
int main()
{
   int i,a[5];
   pthread_t tid[5];

   sem_init(&room,0,4);

   for(i=0;i<5;i++)
      sem_init(&chopstick[i],0,1);
```

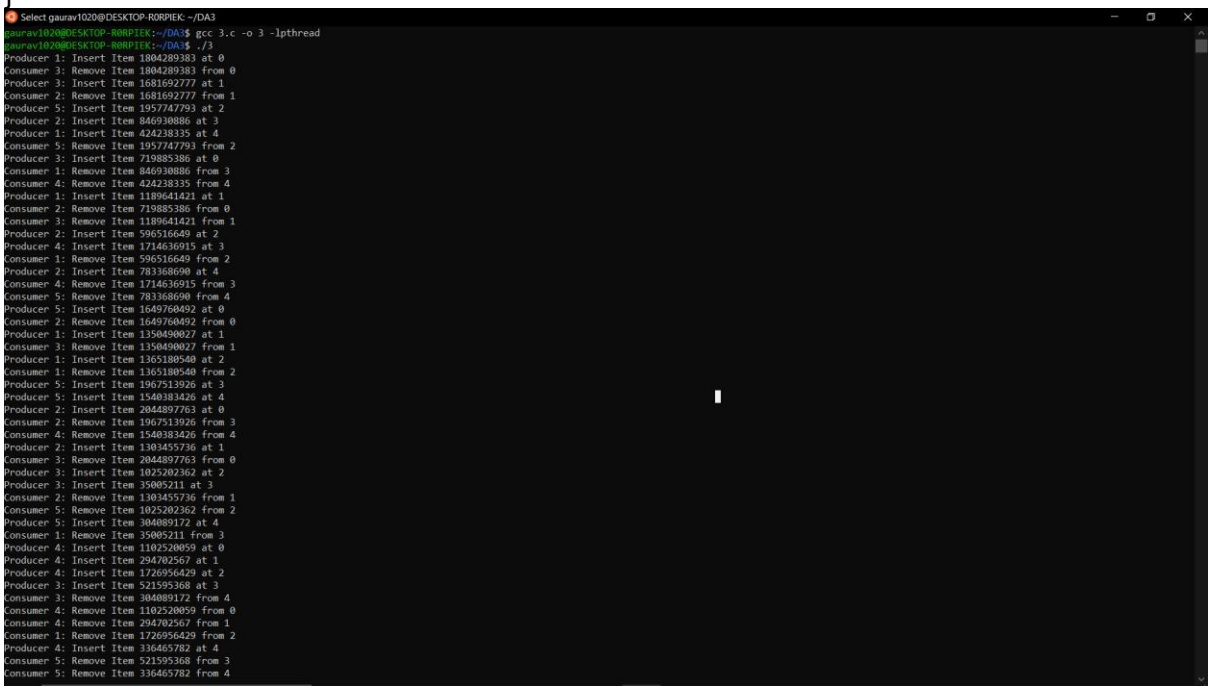
```
    for(i=0;i<5;i++){
        a[i]=i;
        pthread_create(&tid[i],NULL,philosopher,(void *)&a[i]);
    }
    for(i=0;i<5;i++)
        pthread_join(tid[i],NULL);
}

void * philosopher(void * num)
{
    int phil=*(int *)num;

    sem_wait(&room);
    printf("\nPhilosopher %d has entered room",phil);
    sem_wait(&chopstick[phil]);
    sem_wait(&chopstick[(phil+1)%5]);

    eat(phil);
    sleep(2);
    printf("\nPhilosopher %d has finished eating",phil);

    sem_post(&chopstick[(phil+1)%5]);
    sem_post(&chopstick[phil]);
    sem_post(&room);
}

void eat(int phil)
{
    printf("\nPhilosopher %d is eating",phil);
}
```

```
gaurav1020@DESKTOP-R0RPIEK: ~/DA3
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ gcc 2.c -o 2 -lpthread
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ ./2

Philosopher 0 has entered room
Philosopher 0 is eating
Philosopher 2 has entered room
Philosopher 2 is eating
Philosopher 1 has entered room
Philosopher 3 has entered room
Philosopher 0 has finished eating
Philosopher 2 has finished eating
Philosopher 1 is eating
Philosopher 3 is eating
Philosopher 4 has entered room
Philosopher 1 has finished eating
Philosopher 3 has finished eating
Philosopher 4 is eating
Philosopher 4 has finished eatinggaurav1020@DESKTOP-R0RPIEK:~/DA3$
```

(c) Implement the solution for producer consumer problem

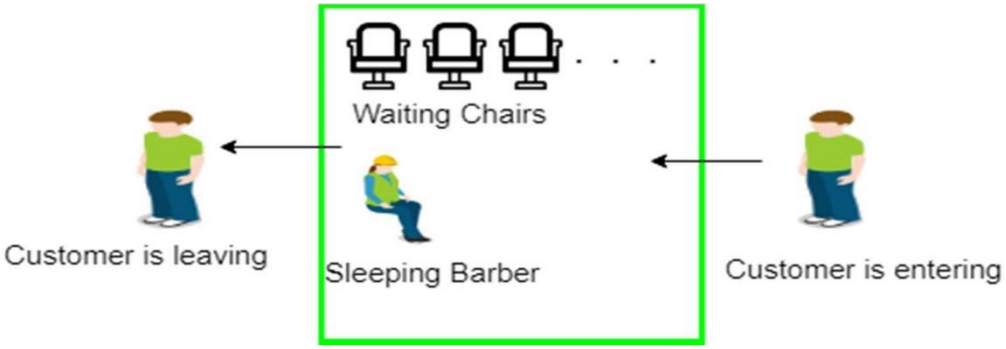
```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#define MaxItems 5
#define BufferSize 5
sem_t empty;
sem_t full;
int in = 0;
int out = 0;
int buffer[BufferSize];
pthread_mutex_t mutex;

void *producer(void *pno)
{
	int item;
	for(int i = 0; i < MaxItems; i++) {
		item = rand(); // Produce an random item
		sem_wait(&empty);
		pthread_mutex_lock(&mutex);
		buffer[in] = item;
		printf("Producer %d: Insert Item %d at %d\n", *((int *)pno),buffer[in],in);
		in = (in+1)%BufferSize;
		pthread_mutex_unlock(&mutex);
		sem_post(&full);
	}
}
void *consumer(void *cno)
{
	for(int i = 0; i < MaxItems; i++) {
		sem_wait(&full);
		pthread_mutex_lock(&mutex);
		int item = buffer[out];
		printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item, out);
		out = (out+1)%BufferSize;
		pthread_mutex_unlock(&mutex);
		sem_post(&empty);
	}
}

int main()
{
	pthread_t pro[5],con[5];
	pthread_mutex_init(&mutex, NULL);
	sem_init(&empty,0,BufferSize);
	sem_init(&full,0,0);
	int a[5] = {1,2,3,4,5}; //Just used for numbering the producer and consumer
	for(int i = 0; i < 5; i++) {
		pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
	}
	for(int i = 0; i < 5; i++) {
```

```
        pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);
    return 0;
}
```



(d) The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barberchair, and n chairs for waiting for customers if there are any to sit on the chair.

- **If there is no customer, then the barber sleeps in his own chair.**
- **When a customer arrives, he has to wake up the barber.**
- **If there are many customers and the barber is cutting a customer'shair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.**



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

#define MAX_CUSTOMERS 25

void *customer(void *num);
void *barber(void *);
void randwait(int secs);

sem_t waitingRoom;
sem_t barberChair;
sem_t barberPillow;
sem_t seatBelt;

int allDone = 0;
int main(int argc, char *argv[]) {
   pthread_t btid;
   pthread_t tid[MAX_CUSTOMERS];
   long RandSeed;
   int i, numCustomers, numChairs;
   int Number[MAX_CUSTOMERS];
   printf("Enter the number of Customers : "); scanf("%d",&numCustomers) ;
   printf("Enter the number of Chairs : "); scanf("%d",&numChairs);
   if (numCustomers > MAX_CUSTOMERS) {
      printf("The maximum number of Customers is %d.\n", MAX_CUSTOMERS);
      exit(-1);
   }
   for (i=0; i<MAX_CUSTOMERS; i++) {
      Number[i] = i;
   }
   sem_init(&waitingRoom, 0, numChairs);
   sem_init(&barberChair, 0, 1);
   sem_init(&barberPillow, 0, 0);
   sem_init(&seatBelt, 0, 0);
   pthread_create(&btid, NULL, barber, NULL);
   for (i=0; i<numCustomers; i++) {
      pthread_create(&tid[i], NULL, customer, (void *)&Number[i]);
      sleep(1);
   }
   for (i=0; i<numCustomers; i++) {
      pthread_join(tid[i],NULL);
      sleep(1);
   }

   allDone = 1;
   sem_post(&barberPillow);
   pthread_join(btid,NULL);
}
void *customer(void *number);
```

```
void *barber(void *junk) {
    while (!allDone) {
        printf("The barber is sleeping\n");
        sem_wait(&barberPillow);
        if (!allDone) {
            printf("The barber is cutting hair\n");
            randwait(2);
            printf("The barber has finished cutting hair.\n");
            sem_post(&seatBelt);
        }
        else {
            printf("The barber is going home for the day.\n");
        }
    }
}
void randwait(int secs) {
    int len;
    len = (int) ((1 * secs) + 1);
    sleep(len);
}

void *customer(void *number) {
    int num = *(int *)number;
    printf("Customer %d leaving for barber shop.\n", num);
    randwait(2);
    printf("Customer %d arrived at barber shop.\n", num);
    sem_wait(&waitingRoom);
    printf("Customer %d entering waiting room.\n", num);
    sem_wait(&barberChair);
    sem_post(&waitingRoom);
    printf("Customer %d waking the barber.\n", num);
    sem_post(&barberPillow);
    sem_wait(&seatBelt);
    sem_post(&barberChair);
    printf("Customer %d leaving barber shop.\n", num);
}
```

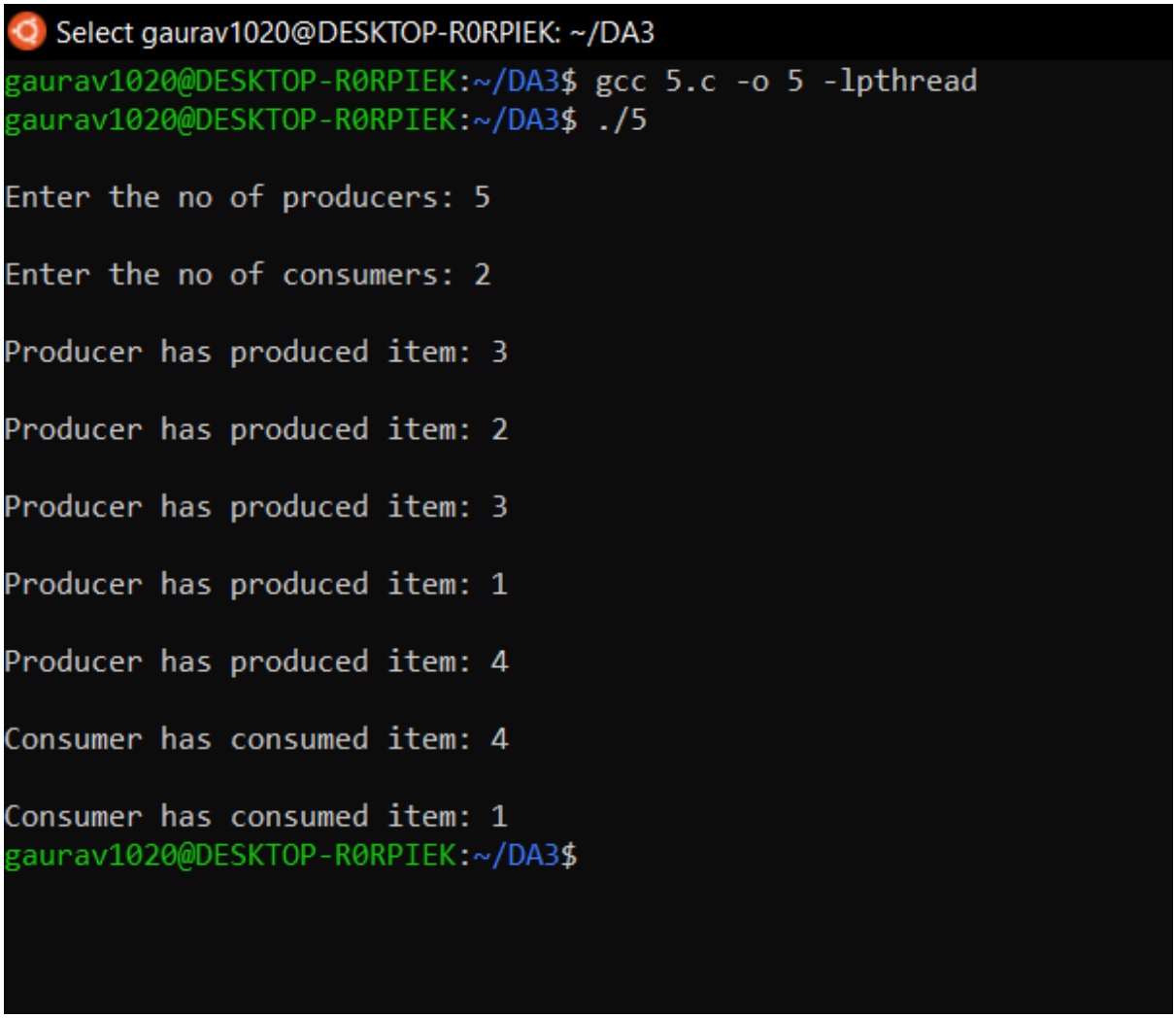
```
gaurav1020@DESKTOP-R0RPIEK: ~/DA3
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ gcc 4.c -o 4 -lpthread
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ ./4
Enter the number of Customers : 3
Enter the number of Chairs : 1
The barber is sleeping
Customer 0 leaving for barber shop.
Customer 1 leaving for barber shop.
Customer 2 leaving for barber shop.
Customer 0 arrived at barber shop.
Customer 0 entering waiting room.
Customer 0 waking the barber.
The barber is cutting hair
Customer 1 arrived at barber shop.
Customer 1 entering waiting room.
Customer 2 arrived at barber shop.
The barber has finished cutting hair.
The barber is sleeping
Customer 0 leaving barber shop.
Customer 1 waking the barber.
Customer 2 entering waiting room.
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 1 leaving barber shop.
Customer 2 waking the barber.
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 2 leaving barber shop.
The barber is going home for the day.
gaurav1020@DESKTOP-R0RPIEK:~/DA3$
```

(e) A pair of processes involved in exchanging a sequence of integers.The number of integers that can be produced and consumed at a time islimited to 100. Write a Program to implement the producer and consumer problem using POSIX semaphore for the above scenario.

```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#include<stdlib.h>
#define buffersize 100
pthread_mutex_t mutex;
pthread_t tidP[100],tidC[100];
```

```
sem_t full,empty;
int counter;
int buffer[buffersize];
void initialize()
{
    pthread_mutex_init(&mutex,NULL);
    sem_init(&full,1,0);
    sem_init(&empty,1,buffersize);
    counter=0;
}
void write(int item)
{
    buffer[counter++]=item;
}
int read()
{
    return(buffer[--counter]);
}
void * producer (void * param)
{
    int waittime,item,i;
    item=rand()%5;
    waittime=rand()%5;
    sem_wait(&empty);pthread_mutex_lock(&mutex);
    printf("\nProducer has produced item: %d\n",item);
    write(item);
    pthread_mutex_unlock(&mutex);
    sem_post(&full);
}
void * consumer (void * param)
{
    int waittime,item;
    waittime=rand()%5;
    sem_wait(&full);
    pthread_mutex_lock(&mutex);
    item=read();
    printf("\nConsumer has consumed item: %d\n",item);
    pthread_mutex_unlock(&mutex);
    sem_post(&empty);
}
int main() {
    int n1, n2, i;
    initialize();
    printf("\nEnter the no of producers: ");
    scanf("%d", &n1);
    printf("\nEnter the no of consumers: ");
    scanf("%d", &n2);
    for (i = 0; i < n1; i++)
        pthread_create(&tidP[i], NULL, producer, NULL);
    for (i = 0; i < n2; i++)
        pthread_create(&tidC[i], NULL, consumer, NULL);
    for (i = 0; i < n1; i++)
        pthread_join(tidP[i], NULL);
    for (i = 0; i < n2; i++)
        pthread_join(tidC[i], NULL);
```

```
    exit(0);
}
```


Select gaurav1020@DESKTOP-R0RPIEK: ~/DA3
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ gcc 5.c -o 5 -lpthread
gaurav1020@DESKTOP-R0RPIEK:~/DA3$ ./5
Enter the no of producers: 5
Enter the no of consumers: 2
Producer has produced item: 3
Producer has produced item: 2
Producer has produced item: 3
Producer has produced item: 1
Producer has produced item: 4
Consumer has consumed item: 4
Consumer has consumed item: 1
gaurav1020@DESKTOP-R0RPIEK:~/DA3$