Kulvir Singh

19BCE2074

CSE2005 – Operating Systems Lab

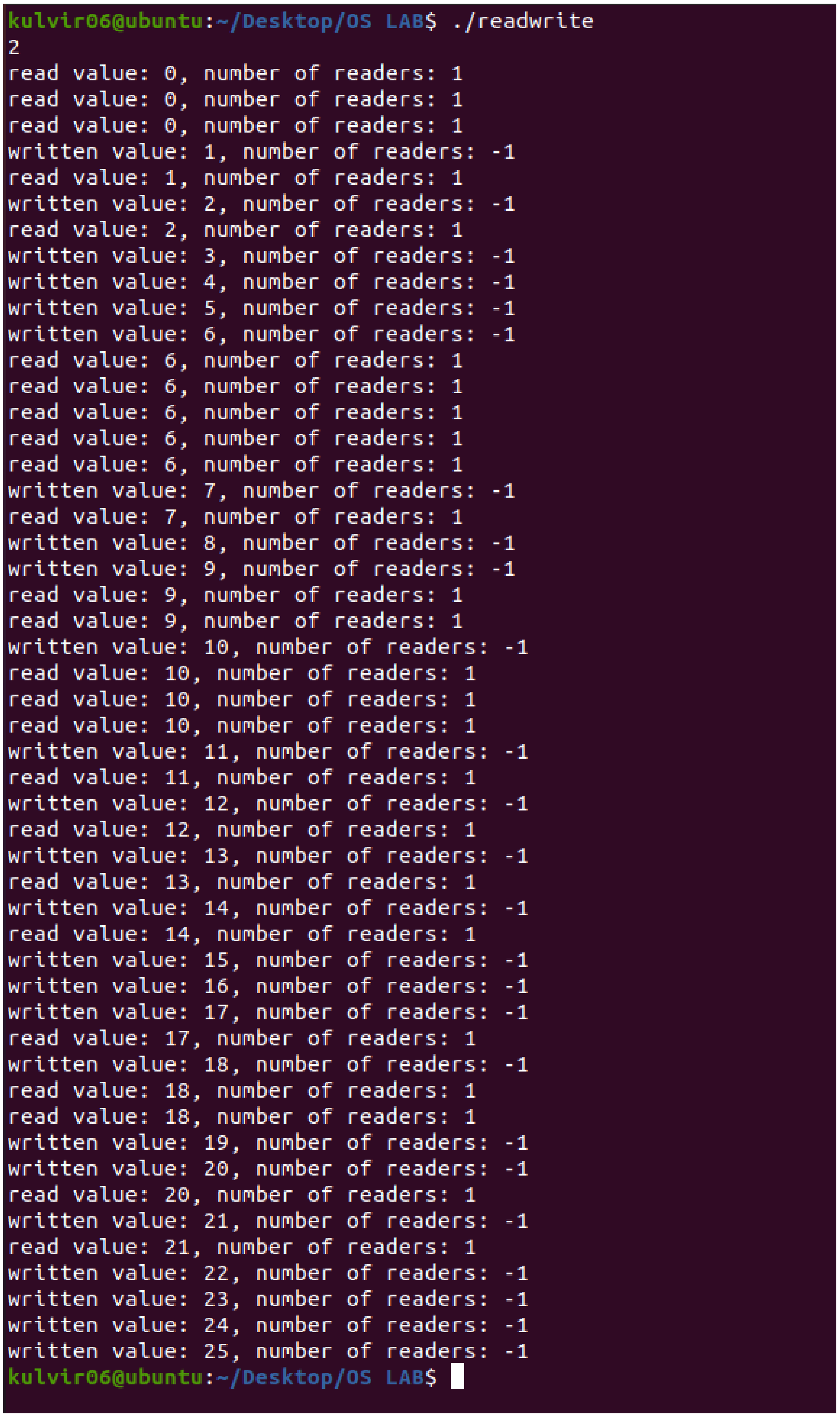
Assessment 3

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

CODE:

```
#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;
}
```

OUTPUT SCREENSHOT


```
kulvir06@ubuntu:~/Desktop/OS LAB$ ./readwrite
2
read value: 0, number of readers: 1
read value: 0, number of readers: 1
read value: 0, number of readers: 1
written value: 1, number of readers: -1
read value: 1, number of readers: 1
written value: 2, number of readers: -1
read value: 2, number of readers: 1
written value: 3, number of readers: -1
written value: 4, number of readers: -1
written value: 5, number of readers: -1
written value: 6, number of readers: -1
read value: 6, number of readers: 1
read value: 6, number of readers: 1
read value: 6, number of readers: 1
read value: 6, number of readers: 1
read value: 6, number of readers: 1
written value: 7, number of readers: -1
read value: 7, number of readers: 1
written value: 8, number of readers: -1
written value: 9, number of readers: -1
read value: 9, number of readers: 1
read value: 9, number of readers: 1
written value: 10, number of readers: -1
read value: 10, number of readers: 1
read value: 10, number of readers: 1
read value: 10, number of readers: 1
written value: 11, number of readers: -1
read value: 11, number of readers: 1
written value: 12, number of readers: -1
read value: 12, number of readers: 1
written value: 13, number of readers: -1
read value: 13, number of readers: 1
written value: 14, number of readers: -1
read value: 14, number of readers: 1
written value: 15, number of readers: -1
written value: 16, number of readers: -1
written value: 17, number of readers: -1
read value: 17, number of readers: 1
written value: 18, number of readers: -1
read value: 18, number of readers: 1
read value: 18, number of readers: 1
written value: 19, number of readers: -1
written value: 20, number of readers: -1
read value: 20, number of readers: 1
written value: 21, number of readers: -1
read value: 21, number of readers: 1
written value: 22, number of readers: -1
written value: 23, number of readers: -1
written value: 24, number of readers: -1
written value: 25, number of readers: -1
kulvir06@ubuntu:~/Desktop/OS LAB$
```

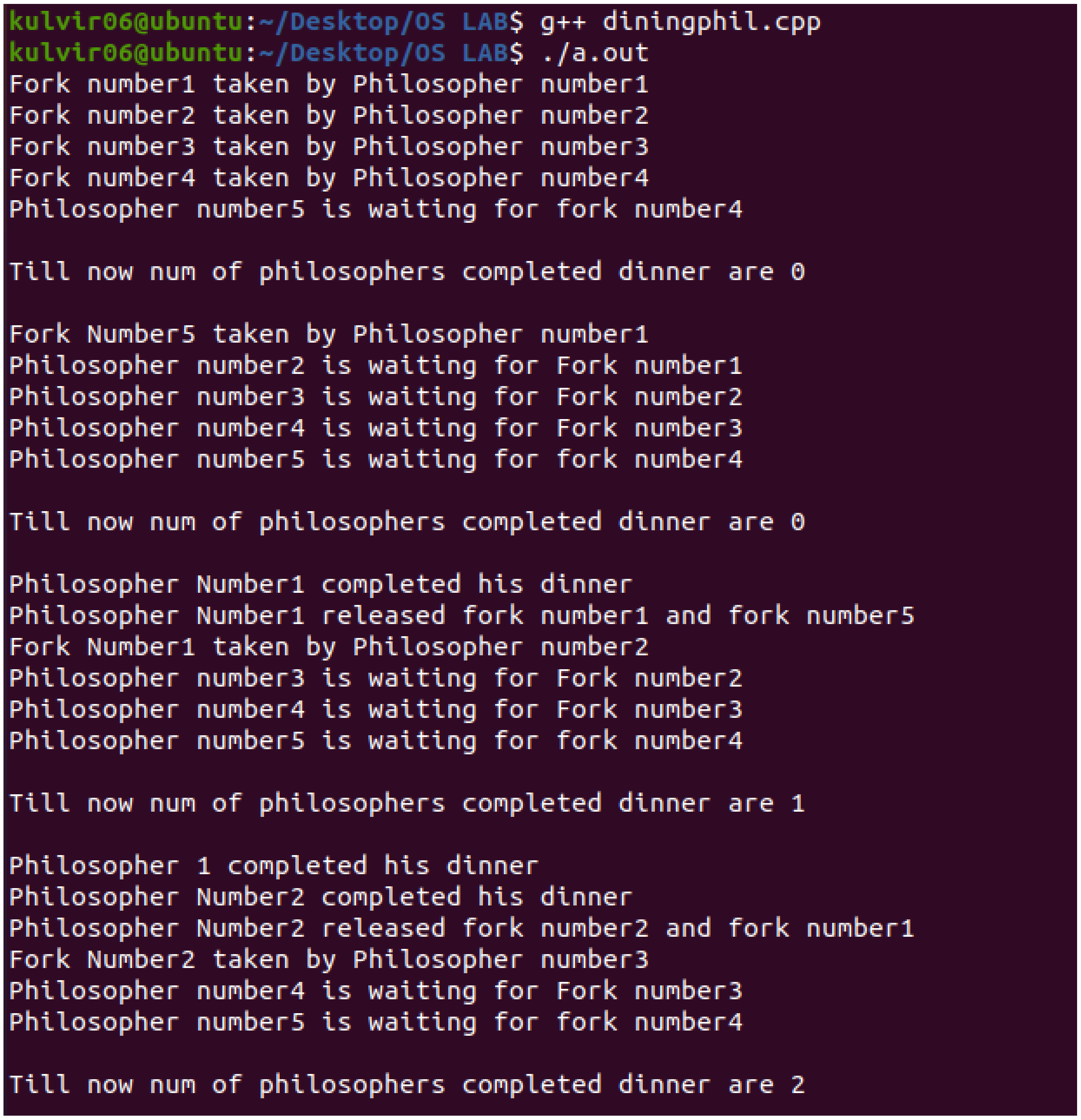
b) Implement the solution for dining philosopher's problem

CODE:

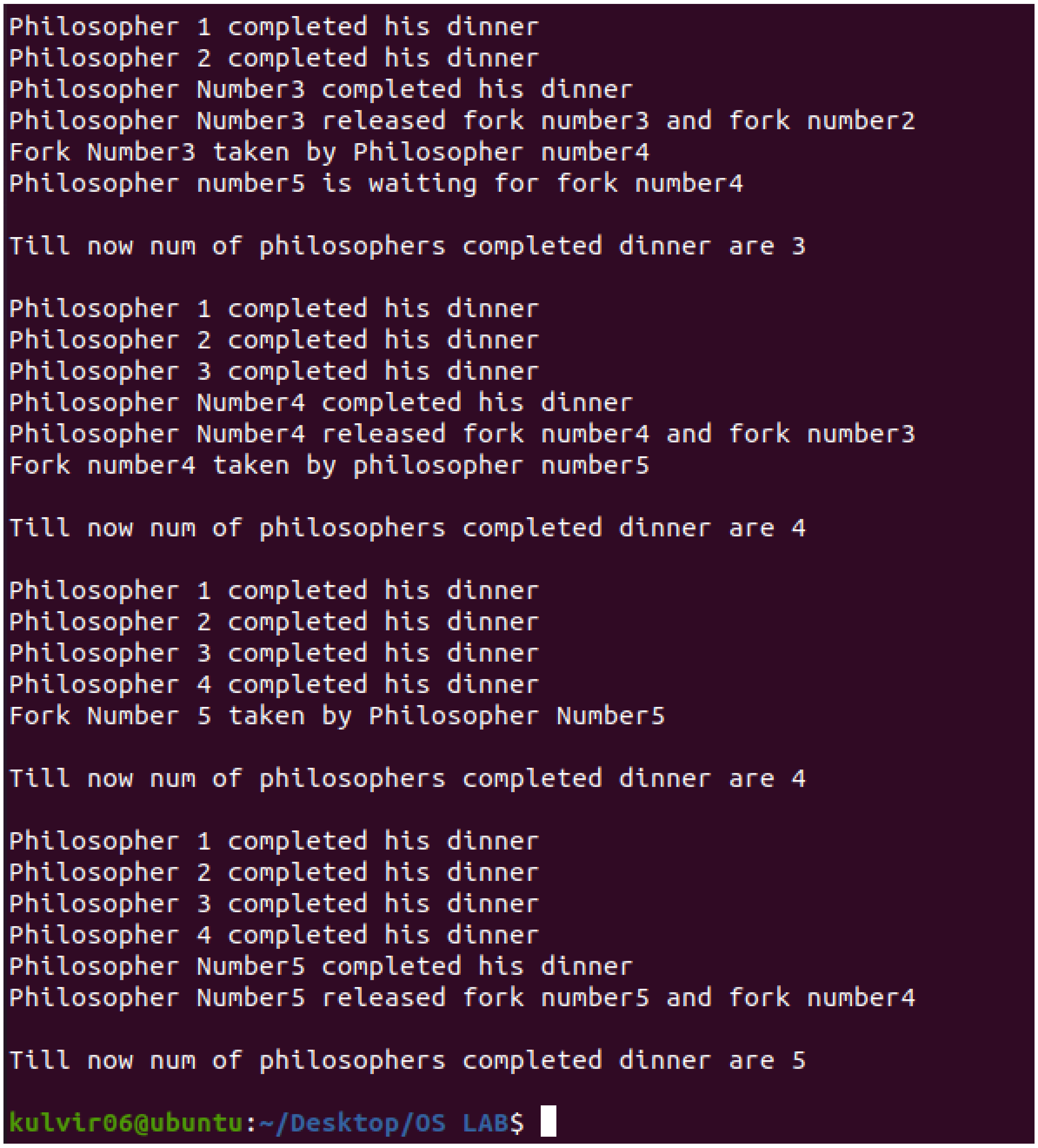
```
#include<iostream>
#define n 5
using namespace std;

int completedPhilosopher = 0,i;

struct fork{
            int takenFork;
}ForkAvailability[n];

struct philosopher{
            int left;
            int right;
}Philostatus[n];

void Dinner(int philosopherID){
            if(Philostatus[philosopherID].left==10 && Philostatus[philosopherID].right==10)
       cout<<"Philosopher "<<philosopherID+1<<" completed his dinner\n";

            else if(Philostatus[philosopherID].left==1 && Philostatus[philosopherID].right==1){

          cout<<"Philosopher Number"<<philosopherID+1<<" completed his dinner\n";

          Philostatus[philosopherID].left = Philostatus[philosopherID].right = 10;
          int fork2 = philosopherID-1;

          if(fork2== -1)
             fork2=(n-1);

          ForkAvailability[philosopherID].takenFork = ForkAvailability[fork2].takenFork = 0;
          cout<<"Philosopher Number"<<philosopherID+1<<" released fork number"<<philosopherID+1<<"
and fork number"<<fork2+1<<"\n";
          completedPhilosopher++;
       }
       else if(Philostatus[philosopherID].left==1 && Philostatus[philosopherID].right==0){
             if(philosopherID==(n-1)){
                if(ForkAvailability[philosopherID].takenFork==0){
                   ForkAvailability[philosopherID].takenFork = Philostatus[philosopherID].right = 1;
                   cout<<"Fork Number "<<philosopherID+1<<" taken by Philosopher
Number"<<philosopherID+1<<"\n";
                }
                else{
                    cout<<"Philosopher Number"<<philosopherID+1<<" is waiting for fork number
"<<philosopherID+1<<"\n";
                }
            }
            else{
                int duplicatephilosopherID = philosopherID;
                philosopherID-=1;

                if(philosopherID== -1)
                    philosopherID=(n-1);

                if(ForkAvailability[philosopherID].takenFork == 0){
                    ForkAvailability[philosopherID].takenFork = Philostatus[duplicatephilosopherID].right = 1;
                    cout<<"Fork Number"<<philosopherID+1<<" taken by Philosopher
number"<<duplicatephilosopherID+1<<"\n";
                }
                else{
                    cout<<"Philosopher number"<<duplicatephilosopherID+1<<" is waiting for Fork
number"<<philosopherID+1<<"\n";
                }
            }
        }
        else if(Philostatus[philosopherID].left==0){
                if(philosopherID==(n-1)){
                    if(ForkAvailability[philosopherID-1].takenFork==0){
                        ForkAvailability[philosopherID-1].takenFork = Philostatus[philosopherID].left = 1;
                        cout<<"Fork number"<<philosopherID<<" taken by philosopher
number"<<philosopherID+1<<"\n";
                    }
                    else{
                        cout<<"Philosopher number"<<philosopherID+1<<" is waiting for fork
number"<<philosopherID<<"\n";
                    }
                }
                else{
                    if(ForkAvailability[philosopherID].takenFork == 0){
                        ForkAvailability[philosopherID].takenFork = Philostatus[philosopherID].left = 1;
                        cout<<"Fork number"<<philosopherID+1<<" taken by Philosopher
number"<<philosopherID+1<<"\n";
                    }
                    else{
                        cout<<"Philosopher number "<<philosopherID+1<<" is waiting for Fork number
"<<philosopherID+1<<"\n";
                    }
                }
    }
}
```

```
int main(){
            for(i=0;i<n;i++)
       ForkAvailability[i].takenFork=Philostatus[i].left=Philostatus[i].right=0;

            while(completedPhilosopher<n){
                        for(i=0;i<n;i++)
          Dinner(i);
                        cout<<"\nTill now num of philosophers completed dinner are
"<<completedPhilosopher<<"\n\n";
            }
            return 0;
}
```

OUTPUT SCREENSHOT



```
kulvir06@ubuntu:~/Desktop/OS LAB$ g++ diningphil.cpp
kulvir06@ubuntu:~/Desktop/OS LAB$ ./a.out
Fork number1 taken by Philosopher number1
Fork number2 taken by Philosopher number2
Fork number3 taken by Philosopher number3
Fork number4 taken by Philosopher number4
Philosopher number5 is waiting for fork number4

Till now num of philosophers completed dinner are 0

Fork Number5 taken by Philosopher number1
Philosopher number2 is waiting for Fork number1
Philosopher number3 is waiting for Fork number2
Philosopher number4 is waiting for Fork number3
Philosopher number5 is waiting for fork number4

Till now num of philosophers completed dinner are 0

Philosopher Number1 completed his dinner
Philosopher Number1 released fork number1 and fork number5
Fork Number1 taken by Philosopher number2
Philosopher number3 is waiting for Fork number2
Philosopher number4 is waiting for Fork number3
Philosopher number5 is waiting for fork number4

Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner
Philosopher Number2 completed his dinner
Philosopher Number2 released fork number2 and fork number1
Fork Number2 taken by Philosopher number3
Philosopher number4 is waiting for Fork number3
Philosopher number5 is waiting for fork number4

Till now num of philosophers completed dinner are 2
```

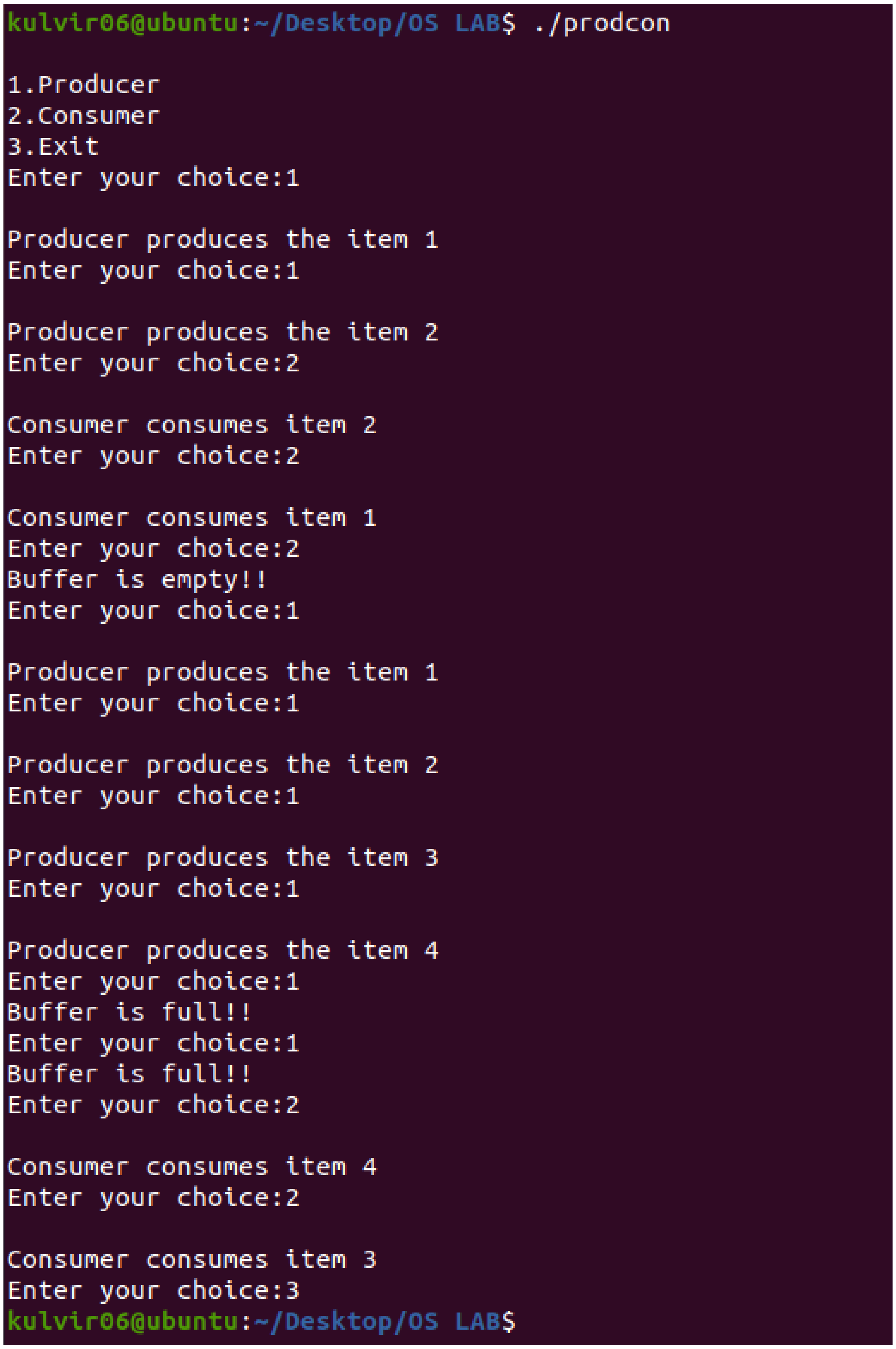

```
Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher Number3 completed his dinner
Philosopher Number3 released fork number3 and fork number2
Fork Number3 taken by Philosopher number4
Philosopher number5 is waiting for fork number4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher Number4 completed his dinner
Philosopher Number4 released fork number4 and fork number3
Fork number4 taken by philosopher number5

Till now num of philosophers completed dinner are 4

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Fork Number 5 taken by Philosopher Number5

Till now num of philosophers completed dinner are 4

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher Number5 completed his dinner
Philosopher Number5 released fork number5 and fork number4

Till now num of philosophers completed dinner are 5

kulvir06@ubuntu:~/Desktop/OS LAB$
```

c) Implement the solution for producer consumer problem

CODE:

```
#include<stdio.h>
#include<stdlib.h>
 int mutex=1,full=0,empty=4,x=0;
 int main()
{
        int n;
        void producer();
        void consumer();
        int wait(int);
        int signal(int);
        printf("\n1.Producer\n2.Consumer\n3.Exit");
        while(1)
        {
                printf("\nEnter your choice:");
                scanf("%d",&n);
                switch(n)
                {
                        case 1:  if((mutex==1)&&(empty!=0))
                                                producer();
                                        else
                                                printf("Buffer is full!!");
                                        break;
                        case 2:  if((mutex==1)&&(full!=0))
                                                consumer();
                                        else
                                                printf("Buffer is empty!!");
                                        break;
                        case 3:
                                        exit(0);
                                        break;
                }
        }

        return 0;
}
int wait(int s)
{
        return (--s);
}

int signal(int s)
{
        return(++s);
}

void producer()
{
	mutex=wait(mutex);
	full=signal(full);
	empty=wait(empty);
	x++;
	printf("\nProducer produces the item %d",x);
	mutex=signal(mutex);
}

void consumer()
{
	mutex=wait(mutex);
	full=wait(full);
	empty=signal(empty);
	printf("\nConsumer consumes item %d",x);
	x--;
	mutex=signal(mutex);
}
```

<u>OUTPUT SCREENSHOT</u>



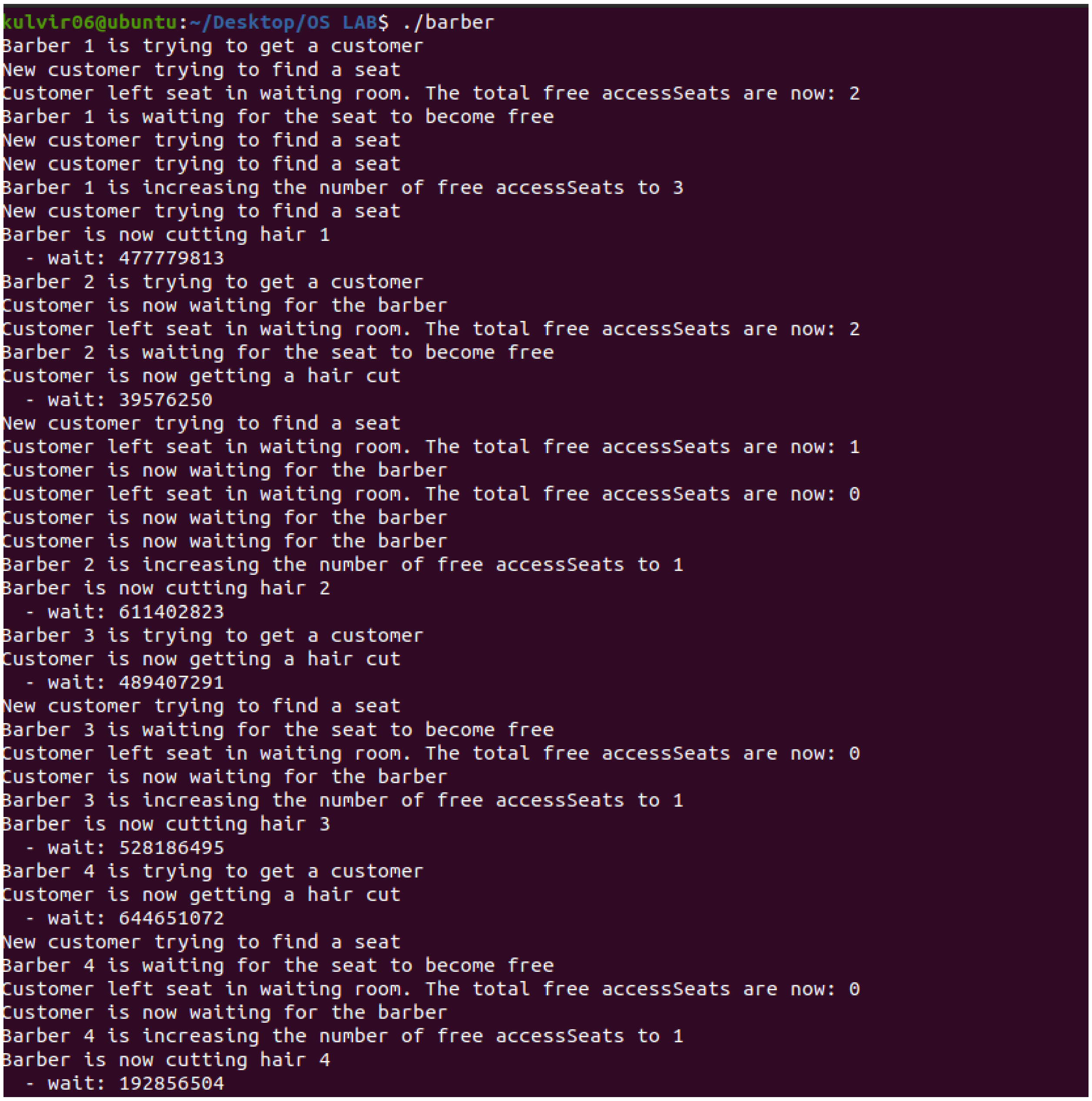
```
kulvir06@ubuntu:~/Desktop/OS LAB$ ./prodcon

1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:1

Producer produces the item 2
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:1

Producer produces the item 1
Enter your choice:1

Producer produces the item 2
Enter your choice:1

Producer produces the item 3
Enter your choice:1

Producer produces the item 4
Enter your choice:1
Buffer is full!!
Enter your choice:1
Buffer is full!!
Enter your choice:2

Consumer consumes item 4
Enter your choice:2

Consumer consumes item 3
Enter your choice:3
kulvir06@ubuntu:~/Desktop/OS LAB$
```

d) The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barber chair, 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's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <time.h>
int accessSeats[2];
int customers[2];
int barber[2];
int freeaccessSeats[2];
void randomWait();
void barber_process();
void customer_process();
void V(int pd[]) {
  int a=1;
  write(pd[1],&a,sizeof(int));
}
void P(int pd[]) {
  int a;
  read(pd[0],&a,sizeof(int));
}
void main() {
  int i;
  pipe(accessSeats);
  pipe(customers);
  pipe(barber);
  pipe(freeaccessSeats);

  V(accessSeats);

  int num = 3; // waiting room seats = 3
  write(freeaccessSeats[1],&num,sizeof(int));
  if (fork() == 0) {
    srand(time(0)+1); //Random Seed
    barber_process();
    return;
  }

  for (i = 1; i <= 5; i++) { // five customers
    if (fork() == 0) {
      srand(time(0)+2*i); // different random seed
      customer_process();
      return;
    }
  }
  sleep(10);
  printf("\ndone\n\n");
}

void barber_process() {
  int i;
  int num; //number of free seats
  for (i = 1; i <= 10; ++i) {
    printf("Barber %d is trying to get a customer\n",i);
    P(customers);
    printf("Barber %d is waiting for the seat to become free\n",i);
    P(accessSeats);
    read(freeaccessSeats[0],&num,sizeof(int));
    num++;
    write(freeaccessSeats[1],&num,sizeof(int));
    printf("Barber %d is increasing the number of free accessSeats to %d\n",i,num);
    V(barber);
    V(accessSeats);
    printf("Barber is now cutting hair %d\n",i);
    randomWait();
  }
}

void customer_process() {
  int i;
  int num;
  for (i = 1; i <= 2; ++i) {
    printf("New customer trying to find a seat\n");
    P(accessSeats);
    read(freeaccessSeats[0],&num,sizeof(int));
    if (num > 0)
    {
      num--;
      write(freeaccessSeats[1],&num,sizeof(int));
      printf("Customer left seat in waiting room. The total free accessSeats are now: %d\n",num);
      V(customers);
      V(accessSeats);
      printf("Customer is now waiting for the barber\n");
      P(barber);
      printf("Customer is now getting a hair cut\n");
    }
    else
    {
      write(freeaccessSeats[1],&num,sizeof(int));
      V(accessSeats);
      printf("No free chairs in waiting room\n");
    }
    randomWait();
  }
}

void randomWait() { // random delay
  int delay;
  delay = random() % 9999999999;
  printf("  - wait: %d\n", delay); // debugging - value of wait time

}
```

OUTPUT SCREENSHOT



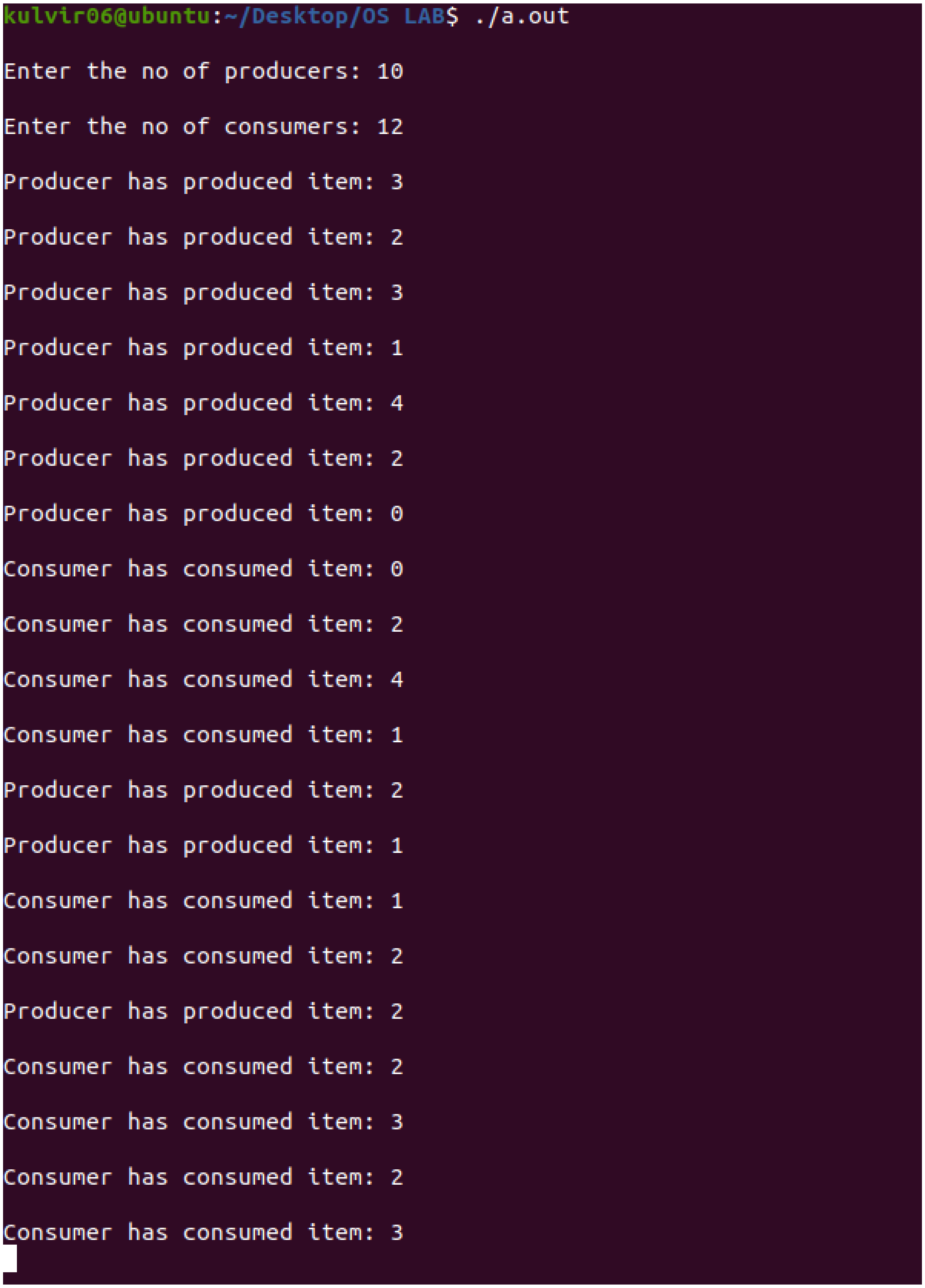
```
kulvir06@ubuntu:~/Desktop/OS LAB$ ./barber
Barber 1 is trying to get a customer
New customer trying to find a seat
Customer left seat in waiting room. The total free accessSeats are now: 2
Barber 1 is waiting for the seat to become free
New customer trying to find a seat
New customer trying to find a seat
Barber 1 is increasing the number of free accessSeats to 3
New customer trying to find a seat
Barber is now cutting hair 1
  - wait: 477779813
Barber 2 is trying to get a customer
Customer is now waiting for the barber
Customer left seat in waiting room. The total free accessSeats are now: 2
Barber 2 is waiting for the seat to become free
Customer is now getting a hair cut
  - wait: 39576250
New customer trying to find a seat
Customer left seat in waiting room. The total free accessSeats are now: 1
Customer is now waiting for the barber
Customer left seat in waiting room. The total free accessSeats are now: 0
Customer is now waiting for the barber
Customer is now waiting for the barber
Barber 2 is increasing the number of free accessSeats to 1
Barber is now cutting hair 2
  - wait: 611402823
Barber 3 is trying to get a customer
Customer is now getting a hair cut
  - wait: 489407291
New customer trying to find a seat
Barber 3 is waiting for the seat to become free
Customer left seat in waiting room. The total free accessSeats are now: 0
Customer is now waiting for the barber
Barber 3 is increasing the number of free accessSeats to 1
Barber is now cutting hair 3
  - wait: 528186495
Barber 4 is trying to get a customer
Customer is now getting a hair cut
  - wait: 644651072
New customer trying to find a seat
Barber 4 is waiting for the seat to become free
Customer left seat in waiting room. The total free accessSeats are now: 0
Customer is now waiting for the barber
Barber 4 is increasing the number of free accessSeats to 1
Barber is now cutting hair 4
  - wait: 192856504
```

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 is limited to 100. Write a Program to implement the producer and consumer problem using POSIX semaphore for the above scenario.

<u>CODE:</u>

```
#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);
//sleep(5);
exit(0);
}
```

OUTPUT SCREENSHOT



```
kulvir06@ubuntu:~/Desktop/OS LAB$ ./a.out

Enter the no of producers: 10

Enter the no of consumers: 12

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

Producer has produced item: 2

Producer has produced item: 0

Consumer has consumed item: 0

Consumer has consumed item: 2

Consumer has consumed item: 4

Consumer has consumed item: 1

Producer has produced item: 2

Producer has produced item: 1

Consumer has consumed item: 1

Consumer has consumed item: 2

Producer has produced item: 2

Consumer has consumed item: 2

Consumer has consumed item: 3

Consumer has consumed item: 2

Consumer has consumed item: 3
```