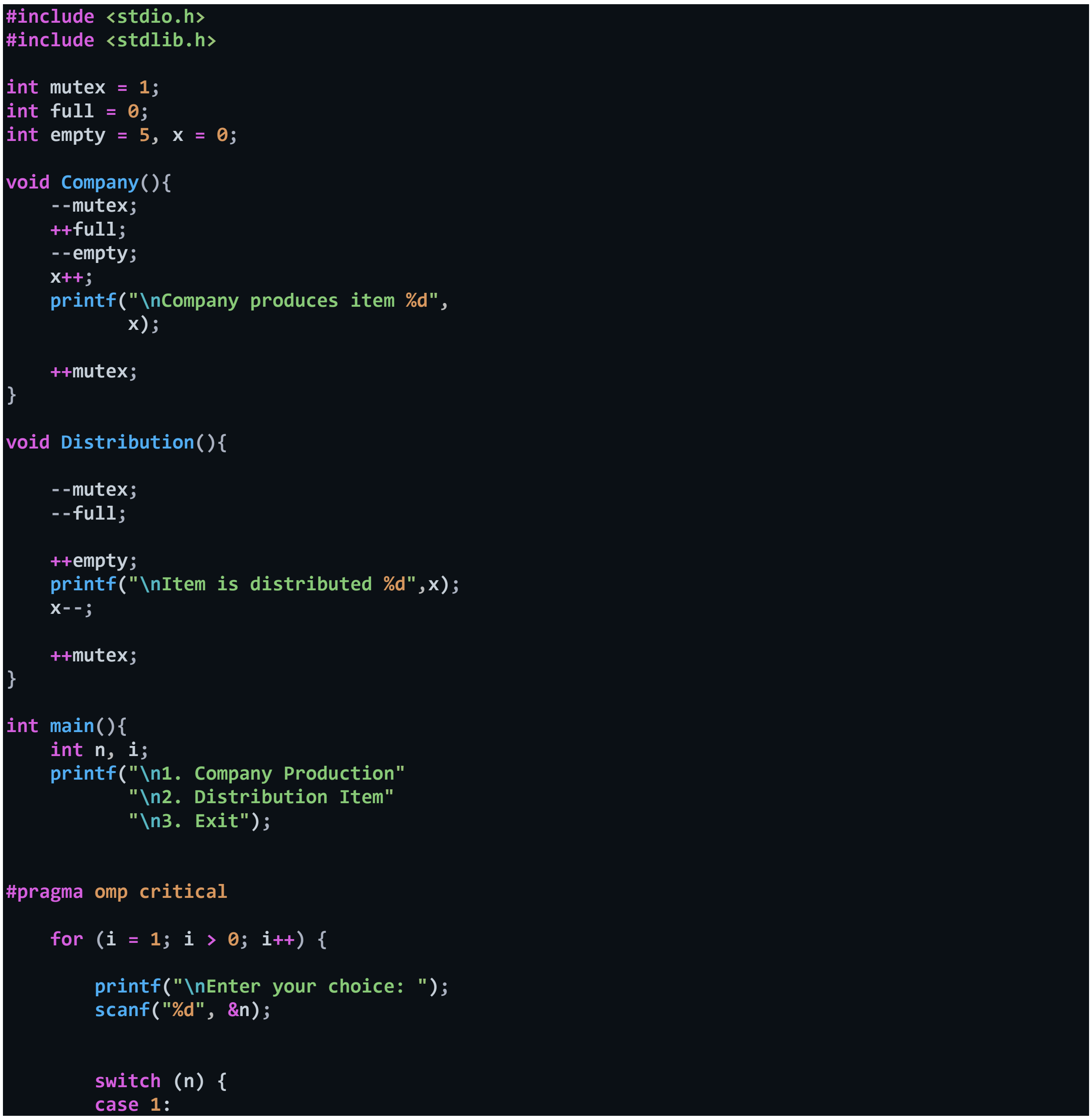
Lab 9:
Synchronization in OS

Programme	:	**BTech. CSE Core**	Semester	:	**Win 2021-22**
Course	:	**Operating Systems**	Code	:	**CSE2005**
Faculty	:	**Dr. Shyamala L**	Slot	:	**L25+L26**
Name	:	**Hariket Sukesh Kumar Sheth**	Register No.	:	**20BCE1975**

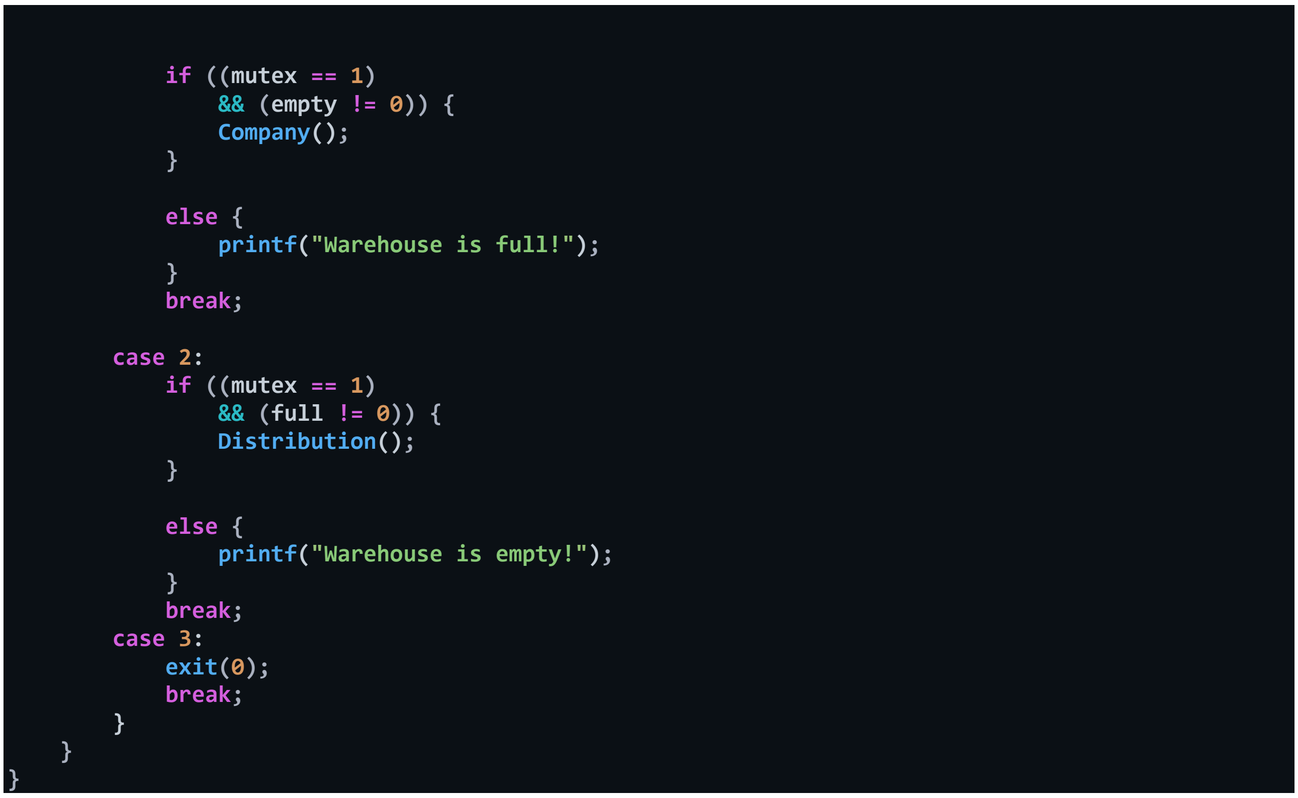
LAB 9

Ques.1 In a Company, goods are manufactured and stored in a sharable warehouse. Goods are distributed to the distributor by the company. Goods can't be manufactured when warehouse is full, and at the same time, Goods can't be distributed when warehouse is empty. Implement the above scenario using appropriate operating system concept.
Write a C-Program to provide a Solution to the above problem.



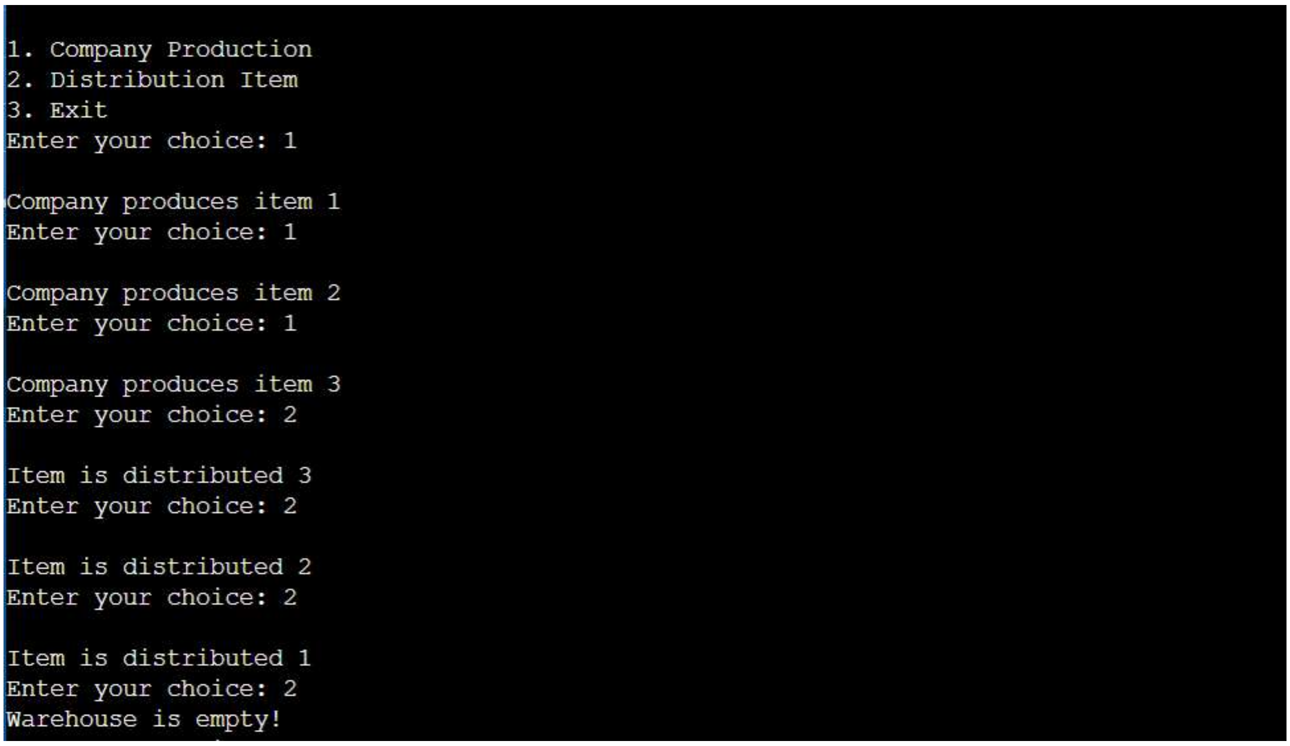
```c
#include <stdio.h>
#include <stdlib.h>

int mutex = 1;
int full = 0;
int empty = 5, x = 0;

void Company(){
    --mutex;
    ++full;
    --empty;
    x++;
    printf("\nCompany produces item %d",
           x);

    ++mutex;
}

void Distribution(){

    --mutex;
    --full;

    ++empty;
    printf("\nItem is distributed %d",x);
    x--;

    ++mutex;
}

int main(){
    int n, i;
    printf("\n1. Company Production"
           "\n2. Distribution Item"
           "\n3. Exit");


#pragma omp critical

    for (i = 1; i > 0; i++) {

        printf("\nEnter your choice: ");
        scanf("%d", &n);


        switch (n) {
        case 1:
```

Name: Hariket Sukesh Kumar Sheth   Register No.: 20BCE1975

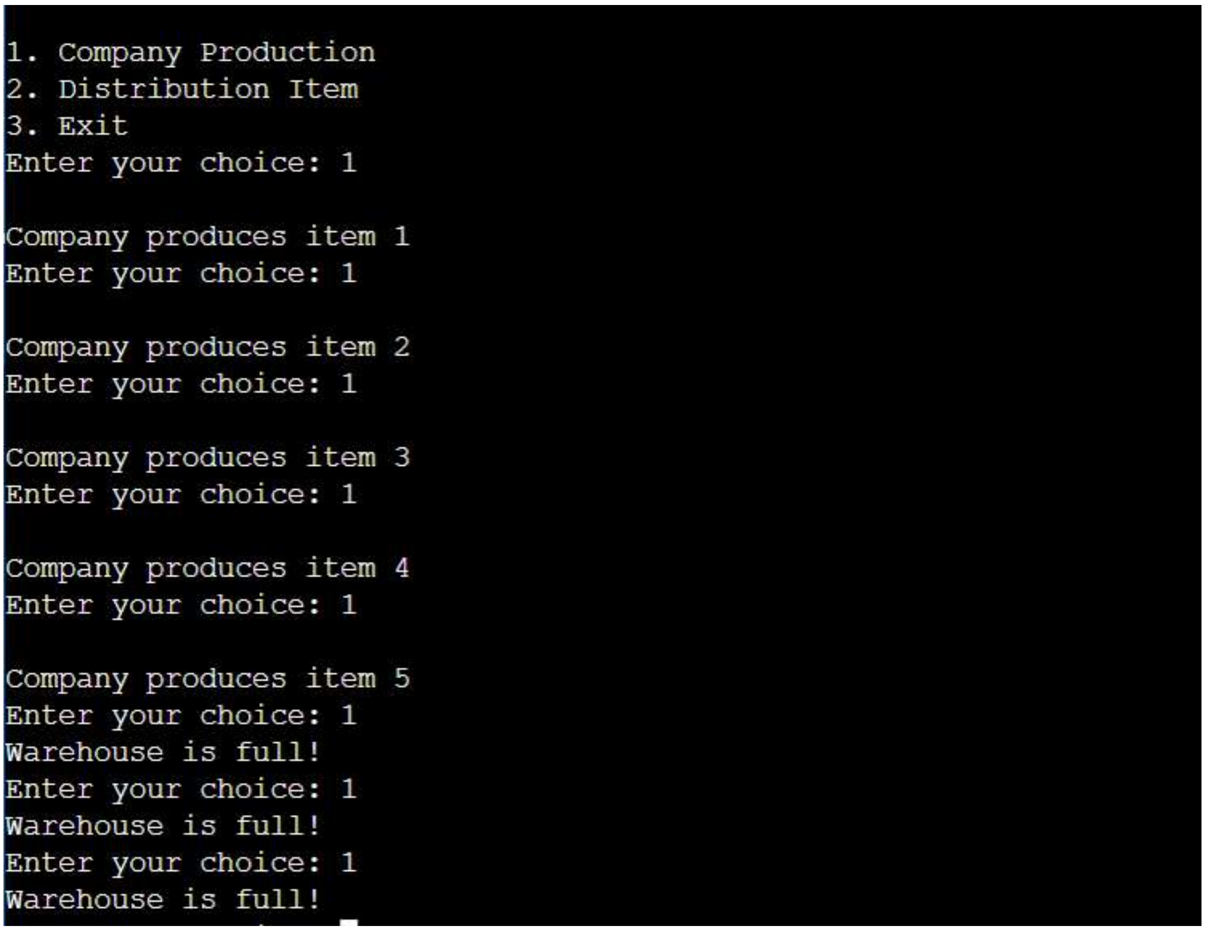


```
            if ((mutex == 1)
                && (empty != 0)) {
                Company();
            }

            else {
                printf("Warehouse is full!");
            }
            break;

        case 2:
            if ((mutex == 1)
                && (full != 0)) {
                Distribution();
            }

            else {
                printf("Warehouse is empty!");
            }
            break;
        case 3:
            exit(0);
            break;
        }
    }
}
```

OUTPUT:

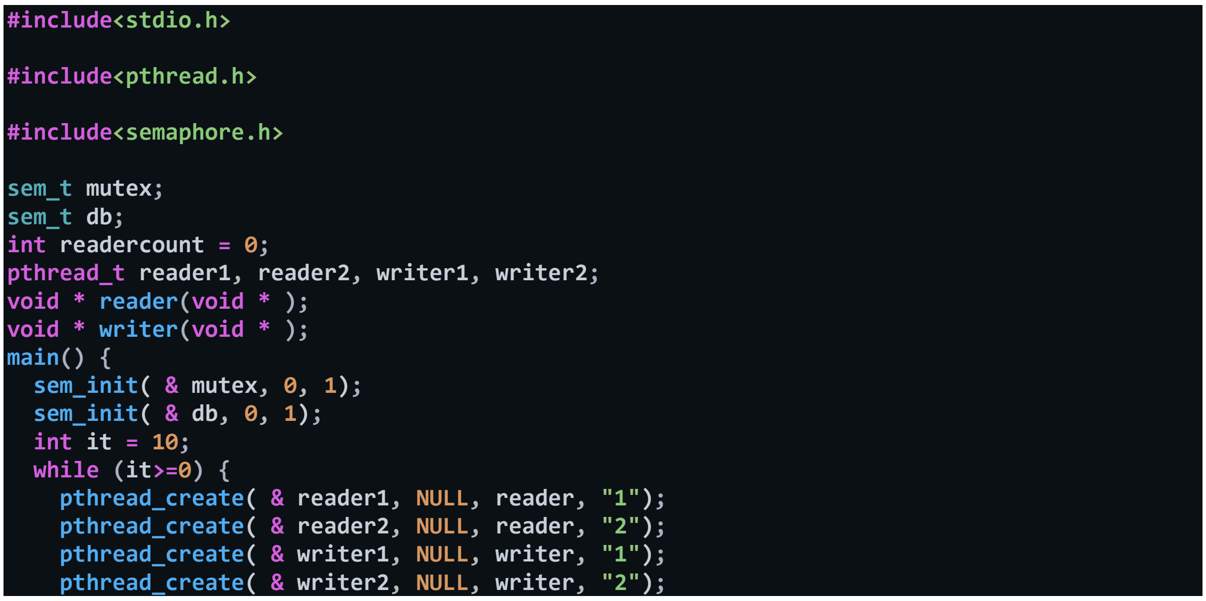


```
1. Company Production
2. Distribution Item
3. Exit
Enter your choice: 1

Company produces item 1
Enter your choice: 1

Company produces item 2
Enter your choice: 1

Company produces item 3
Enter your choice: 2

Item is distributed 3
Enter your choice: 2

Item is distributed 2
Enter your choice: 2

Item is distributed 1
Enter your choice: 2
Warehouse is empty!
```



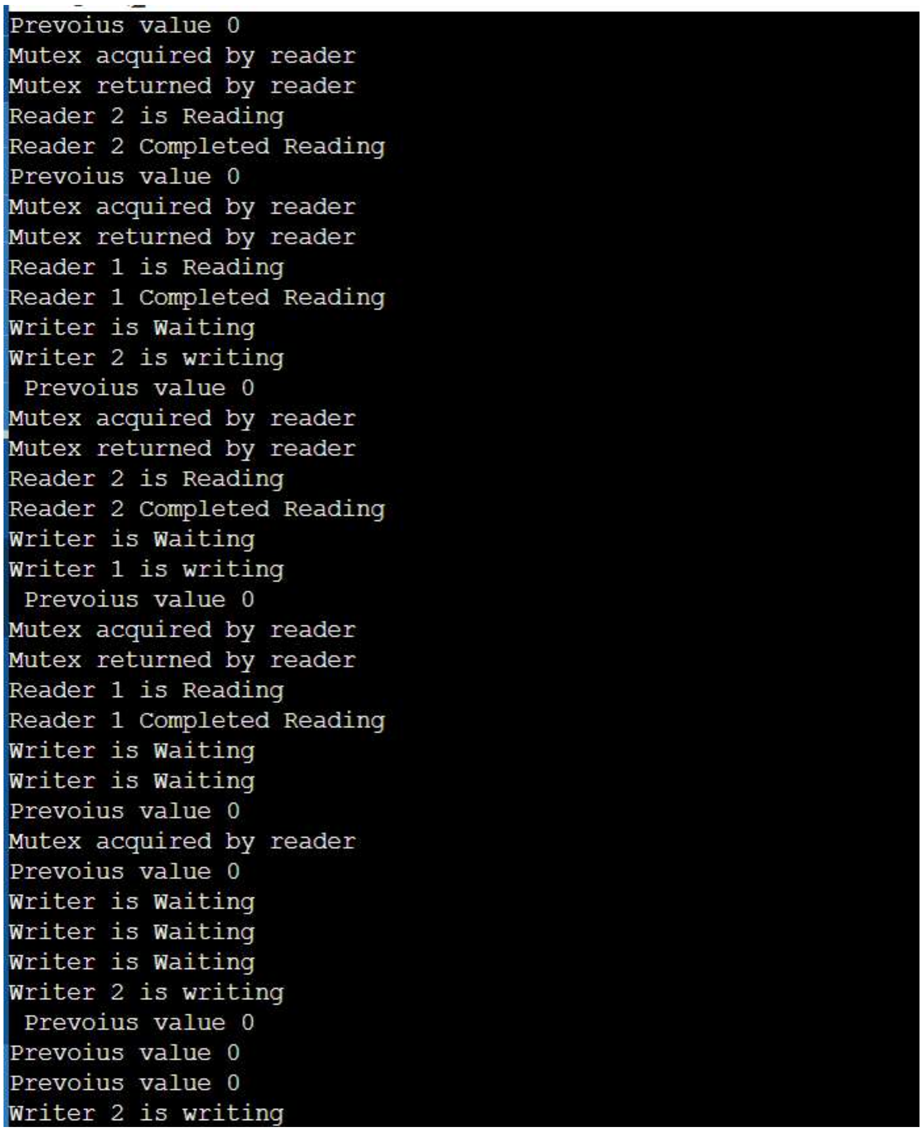
```
1. Company Production
2. Distribution Item
3. Exit
Enter your choice: 1

Company produces item 1
Enter your choice: 1

Company produces item 2
Enter your choice: 1

Company produces item 3
Enter your choice: 1

Company produces item 4
Enter your choice: 1

Company produces item 5
Enter your choice: 1
Warehouse is full!
Enter your choice: 1
Warehouse is full!
Enter your choice: 1
Warehouse is full!
```

Ques.2 We have a database which many users are allowed to read simultaneously, but to which only one user may write at a time (to avoid race conditions in the database). So it is identical to the mutual exclusion problem, except that we wish to permit more flexibility in the system by letting more than one simultaneous reader. Naturally there is no danger of a race condition when many tasks are *reading* from a shared data structure.
Write a C-Program to provide a Solution to the above problem.



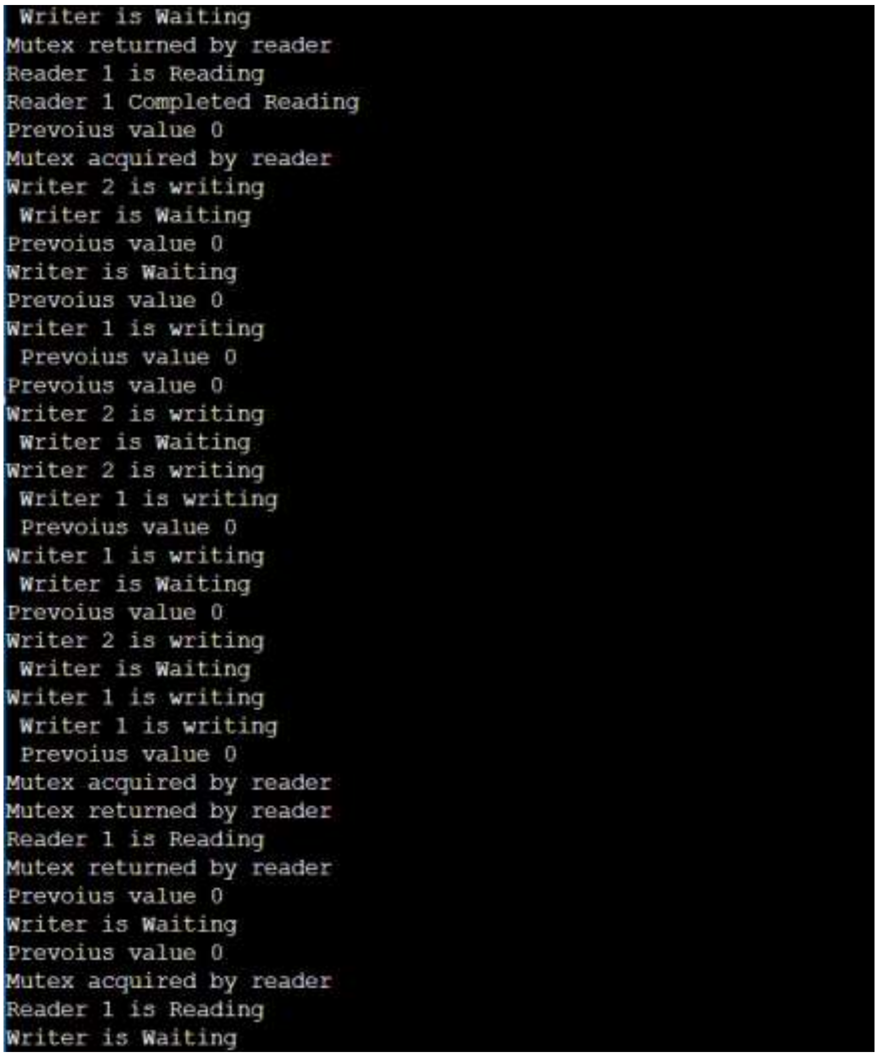
```
#include<stdio.h>

#include<pthread.h>

#include<semaphore.h>

sem_t mutex;
sem_t db;
int readercount = 0;
pthread_t reader1, reader2, writer1, writer2;
void * reader(void * );
void * writer(void * );
main() {
  sem_init( & mutex, 0, 1);
  sem_init( & db, 0, 1);
  int it = 10;
  while (it>=0) {
    pthread_create( & reader1, NULL, reader, "1");
    pthread_create( & reader2, NULL, reader, "2");
    pthread_create( & writer1, NULL, writer, "1");
    pthread_create( & writer2, NULL, writer, "2");
    it--;
  }
}
void * reader(void * p) {
  printf("Prevoius value %d\n", mutex);
  sem_wait( & mutex);
  printf("Mutex acquired by reader\n", mutex);
  readercount++;
  if (readercount == 1) sem_wait( & db);
  sem_post( & mutex);
  printf("Mutex returned by reader\n", mutex);
  printf("Reader %s is Reading\n", p);
  sem_wait( & mutex);
  printf("Reader %s Completed Reading\n", p);
  readercount--;
  if (readercount == 0) sem_post( & db);
  sem_post( & mutex);
}

void * writer(void * p) {
  printf("Writer is Waiting \n");
  sem_wait( & db);
  printf("Writer %s is writing\n ", p);
  sem_post( & db);
}
```

OUTPUT:



```
Prevoius value 0
Mutex acquired by reader
Mutex returned by reader
Reader 2 is Reading
Reader 2 Completed Reading
Prevoius value 0
Mutex acquired by reader
Mutex returned by reader
Reader 1 is Reading
Reader 1 Completed Reading
Writer is Waiting
Writer 2 is writing
 Prevoius value 0
Mutex acquired by reader
Mutex returned by reader
Reader 2 is Reading
Reader 2 Completed Reading
Writer is Waiting
Writer 1 is writing
 Prevoius value 0
Mutex acquired by reader
Mutex returned by reader
Reader 1 is Reading
Reader 1 Completed Reading
Writer is Waiting
Writer is Waiting
Prevoius value 0
Mutex acquired by reader
Prevoius value 0
Writer is Waiting
Writer is Waiting
Writer is Waiting
Writer 2 is writing
 Prevoius value 0
Prevoius value 0
Prevoius value 0
Writer 2 is writing
```

Name: Hariket Sukesh Kumar Sheth Register No.: 20BCE1975



```
 Writer is Waiting
Mutex returned by reader
Reader 1 is Reading
Reader 1 Completed Reading
Prevoius value 0
Mutex acquired by reader
Writer 2 is writing
 Writer is Waiting
Prevoius value 0
Writer is Waiting
Prevoius value 0
Writer 1 is writing
 Prevoius value 0
Prevoius value 0
Writer 2 is writing
 Writer is Waiting
Writer 2 is writing
 Writer 1 is writing
 Prevoius value 0
Writer 1 is writing
 Writer is Waiting
Prevoius value 0
Writer 2 is writing
 Writer is Waiting
Writer 1 is writing
 Writer 1 is writing
 Prevoius value 0
Mutex acquired by reader
Mutex returned by reader
Reader 1 is Reading
Mutex returned by reader
Prevoius value 0
Writer is Waiting
Prevoius value 0
Mutex acquired by reader
Reader 1 is Reading
Writer is Waiting
```