National Textile University

Department of Computer Science

Subject:

Operating system

Submitted To:

Sir Nasir

Submitted By:

Noor ul Ain

Registration No:

23-NTU-CS-1221

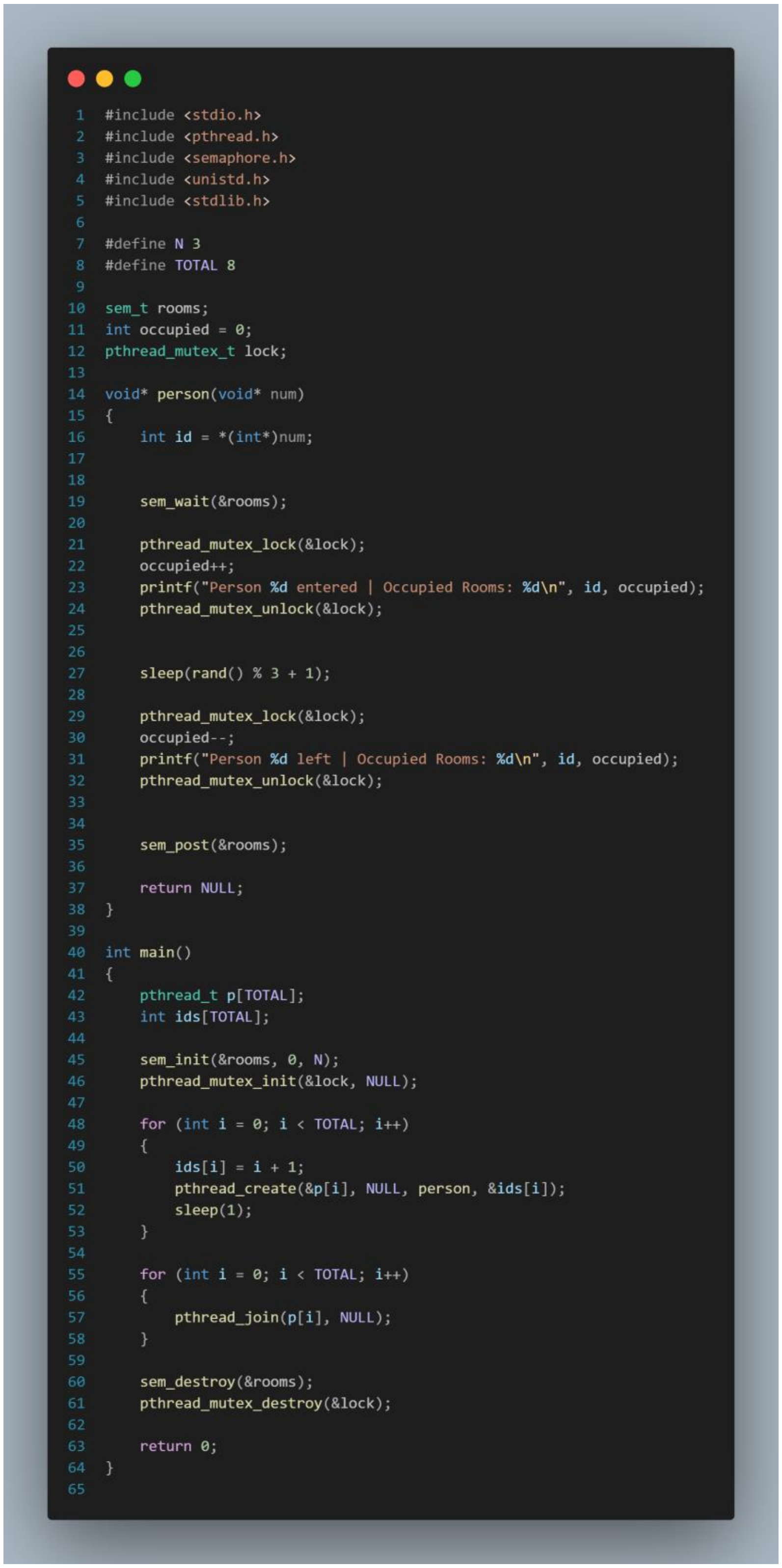
Home Task No:

10

Semester:

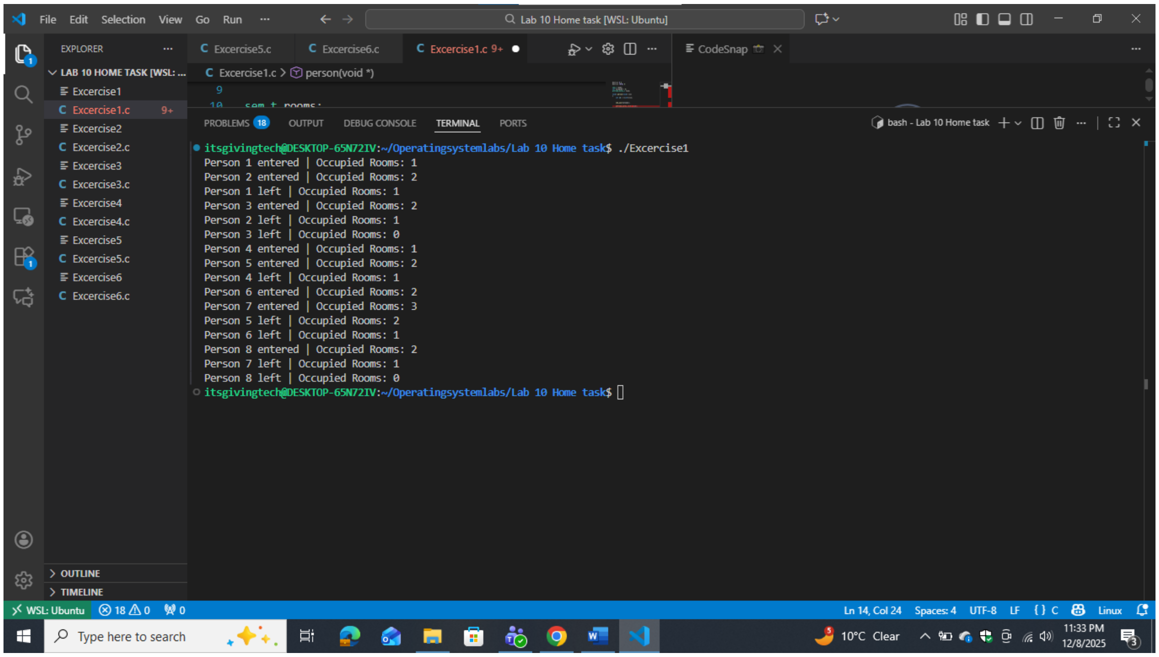
5th

Exercise 1:



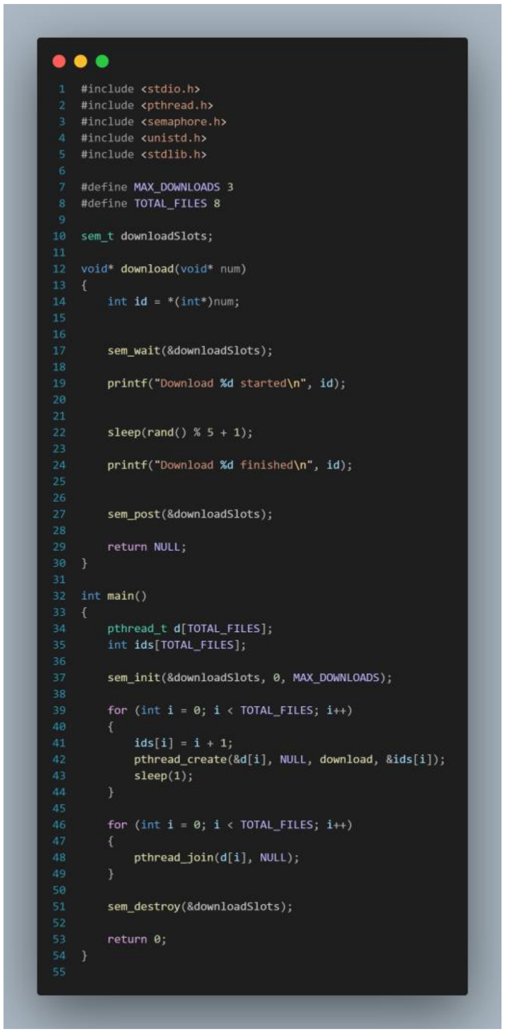
```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdlib.h>

#define N 3
#define TOTAL 8

sem_t rooms;
int occupied = 0;
pthread_mutex_t lock;

void* person(void* num)
{
    int id = *(int*)num;


    sem_wait(&rooms);

    pthread_mutex_lock(&lock);
    occupied++;
    printf("Person %d entered | Occupied Rooms: %d\n", id, occupied);
    pthread_mutex_unlock(&lock);


    sleep(rand() % 3 + 1);

    pthread_mutex_lock(&lock);
    occupied--;
    printf("Person %d left | Occupied Rooms: %d\n", id, occupied);
    pthread_mutex_unlock(&lock);


    sem_post(&rooms);

    return NULL;
}

int main()
{
    pthread_t p[TOTAL];
    int ids[TOTAL];

    sem_init(&rooms, 0, N);
    pthread_mutex_init(&lock, NULL);

    for (int i = 0; i < TOTAL; i++)
    {
        ids[i] = i + 1;
        pthread_create(&p[i], NULL, person, &ids[i]);
        sleep(1);
    }

    for (int i = 0; i < TOTAL; i++)
    {
        pthread_join(p[i], NULL);
    }

    sem_destroy(&rooms);
    pthread_mutex_destroy(&lock);

    return 0;
}
```

Output:



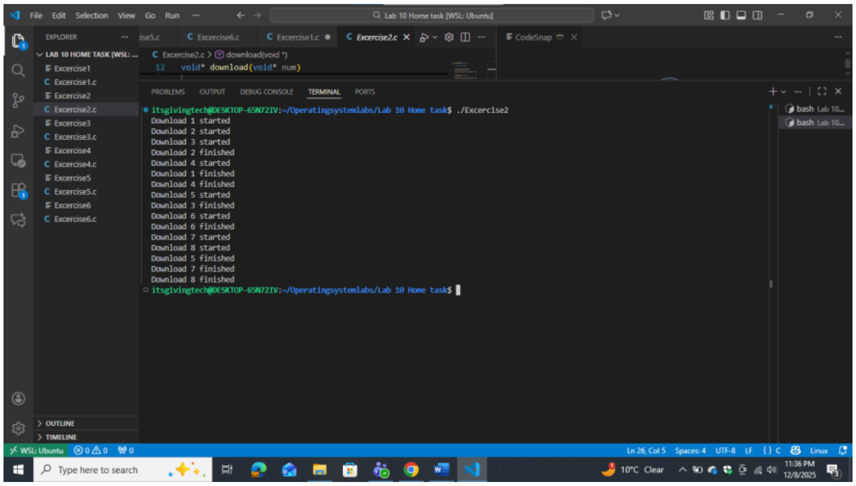
```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise1
Person 1 entered | Occupied Rooms: 1
Person 2 entered | Occupied Rooms: 2
Person 1 left | Occupied Rooms: 1
Person 3 entered | Occupied Rooms: 2
Person 2 left | Occupied Rooms: 1
Person 3 left | Occupied Rooms: 0
Person 4 entered | Occupied Rooms: 1
Person 5 entered | Occupied Rooms: 2
Person 4 left | Occupied Rooms: 1
Person 6 entered | Occupied Rooms: 2
Person 7 entered | Occupied Rooms: 3
Person 5 left | Occupied Rooms: 2
Person 6 left | Occupied Rooms: 1
Person 8 entered | Occupied Rooms: 2
Person 7 left | Occupied Rooms: 1
Person 8 left | Occupied Rooms: 0
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ 
```

Exercise 2:



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

#define MAX_DOWNLOADS 3
#define TOTAL_FILES 8

sem_t downloadSlots;

void* download(void* num)
{
    int id = *(int*)num;


    sem_wait(&downloadSlots);

    printf("Download %d started\n", id);


    sleep(rand() % 5 + 1);

    printf("Download %d finished\n", id);


    sem_post(&downloadSlots);

    return NULL;
}

int main()
{
    pthread_t d[TOTAL_FILES];
    int ids[TOTAL_FILES];

    sem_init(&downloadSlots, 0, MAX_DOWNLOADS);

    for (int i = 0; i < TOTAL_FILES; i++)
    {
        ids[i] = i + 1;
        pthread_create(&d[i], NULL, download, &ids[i]);
        sleep(1);
    }

    for (int i = 0; i < TOTAL_FILES; i++)
    {
        pthread_join(d[i], NULL);
    }

    sem_destroy(&downloadSlots);

    return 0;
}
```

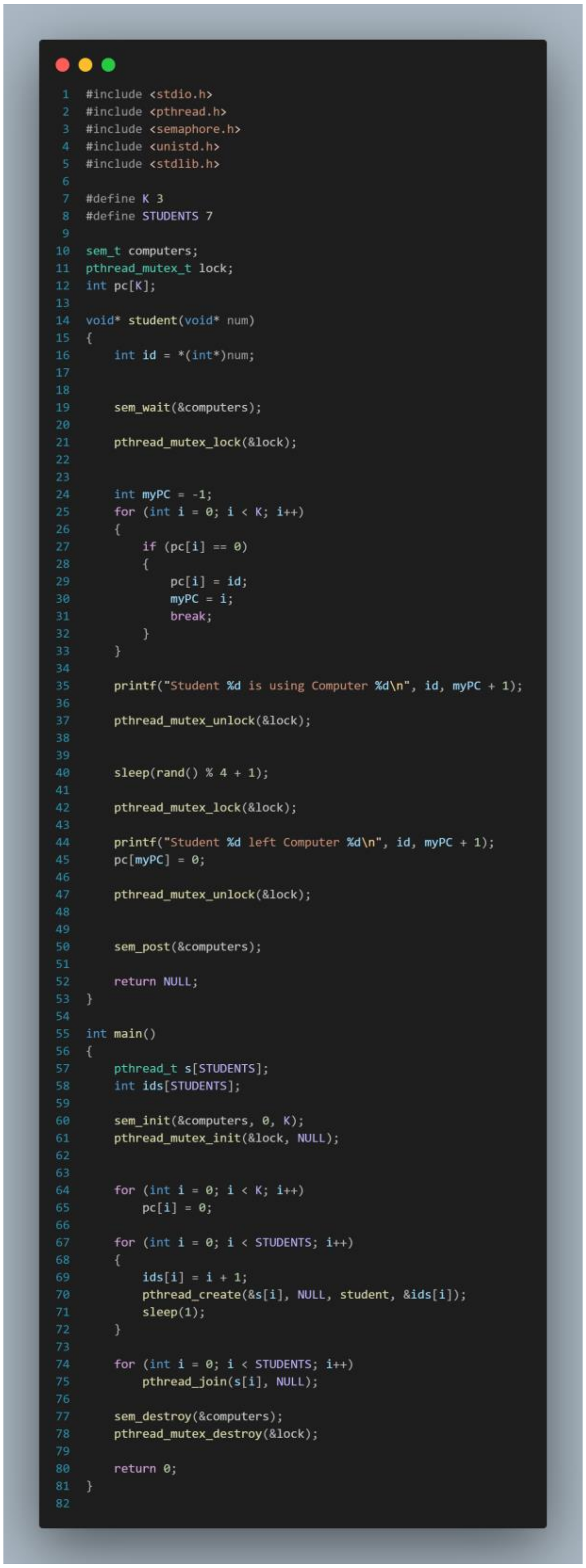
Output:



```
void* download(void* num)
```

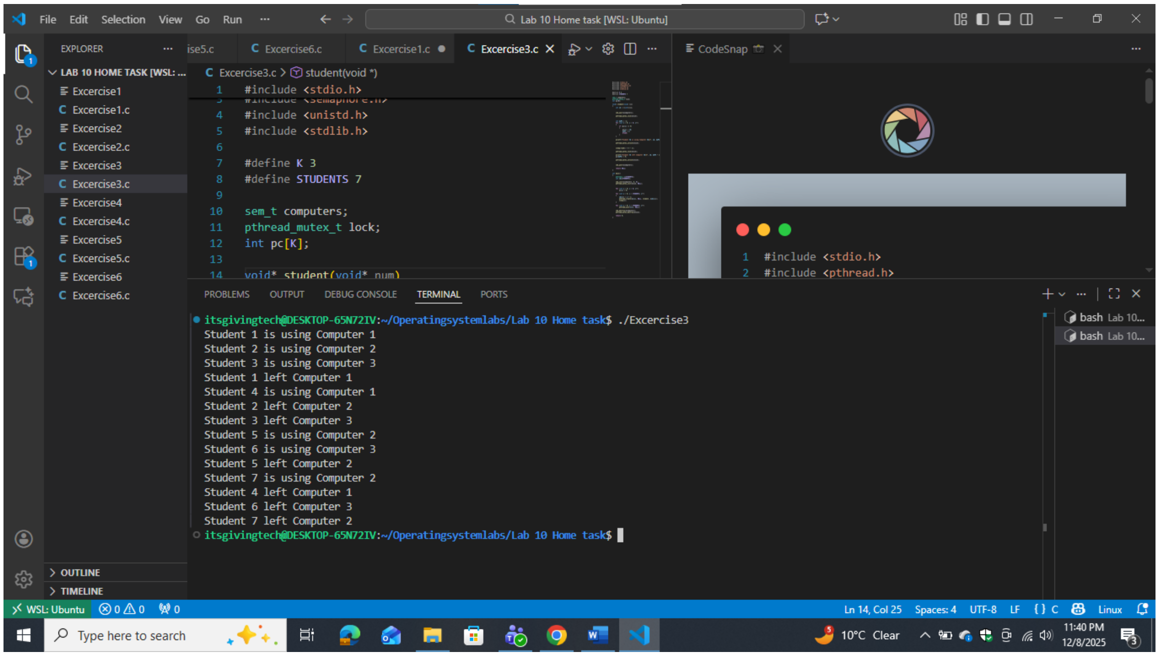
```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise2
Download 1 started
Download 2 started
Download 3 started
Download 2 finished
Download 4 started
Download 1 finished
Download 4 finished
Download 5 started
Download 3 finished
Download 6 started
Download 6 finished
Download 7 started
Download 8 started
Download 5 finished
Download 7 finished
Download 8 finished
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$
```

Exercise 3:



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

#define K 3
#define STUDENTS 7

sem_t computers;
pthread_mutex_t lock;
int pc[K];

void* student(void* num)
{
    int id = *(int*)num;


    sem_wait(&computers);

    pthread_mutex_lock(&lock);


    int myPC = -1;
    for (int i = 0; i < K; i++)
    {
        if (pc[i] == 0)
        {
            pc[i] = id;
            myPC = i;
            break;
        }
    }

    printf("Student %d is using Computer %d\n", id, myPC + 1);

    pthread_mutex_unlock(&lock);


    sleep(rand() % 4 + 1);

    pthread_mutex_lock(&lock);

    printf("Student %d left Computer %d\n", id, myPC + 1);
    pc[myPC] = 0;

    pthread_mutex_unlock(&lock);


    sem_post(&computers);

    return NULL;
}

int main()
{
    pthread_t s[STUDENTS];
    int ids[STUDENTS];

    sem_init(&computers, 0, K);
    pthread_mutex_init(&lock, NULL);


    for (int i = 0; i < K; i++)
        pc[i] = 0;

    for (int i = 0; i < STUDENTS; i++)
    {
        ids[i] = i + 1;
        pthread_create(&s[i], NULL, student, &ids[i]);
        sleep(1);
    }

    for (int i = 0; i < STUDENTS; i++)
        pthread_join(s[i], NULL);

    sem_destroy(&computers);
    pthread_mutex_destroy(&lock);

    return 0;
}

```

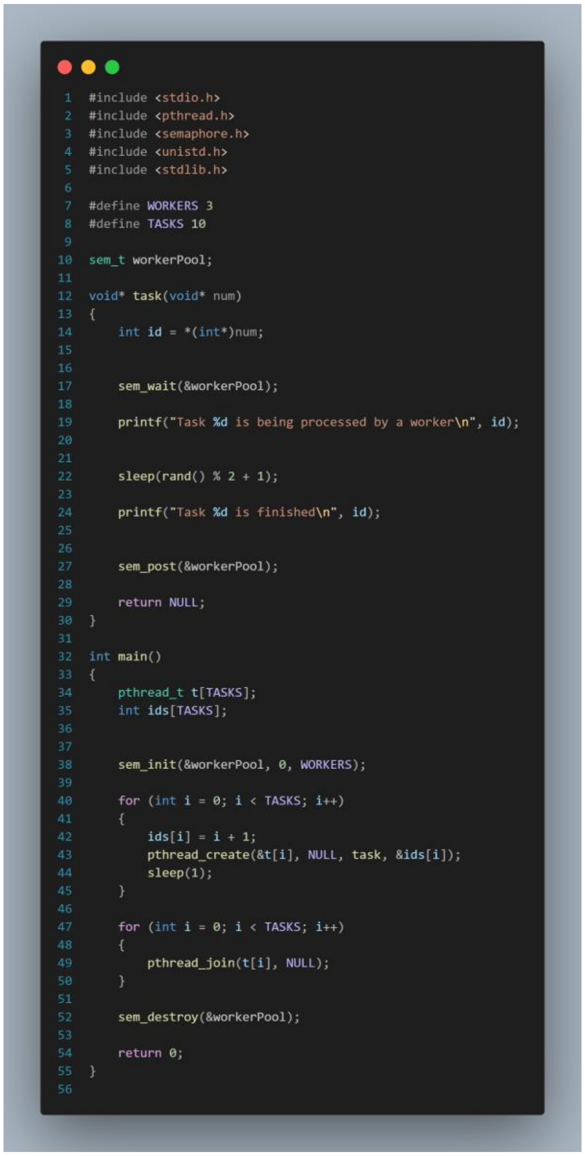
Output:



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

#define K 3
#define STUDENTS 7

sem_t computers;
pthread_mutex_t lock;
int pc[K];

void* student(void* num)
```

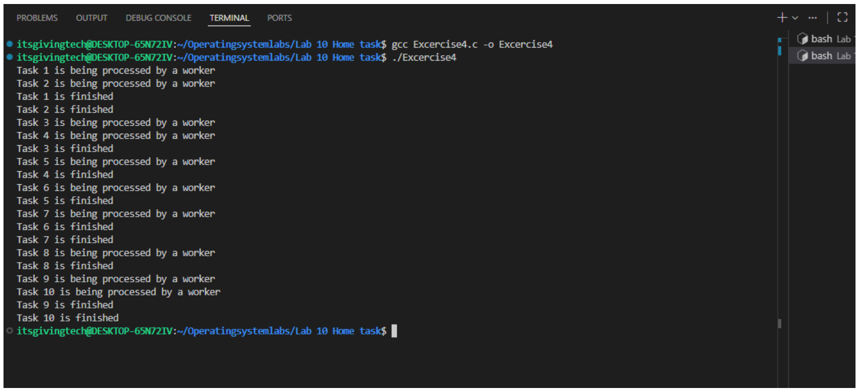
```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise3
Student 1 is using Computer 1
Student 2 is using Computer 2
Student 3 is using Computer 3
Student 1 left Computer 1
Student 4 is using Computer 1
Student 2 left Computer 2
Student 3 left Computer 3
Student 5 is using Computer 2
Student 6 is using Computer 3
Student 5 left Computer 2
Student 7 is using Computer 2
Student 4 left Computer 1
Student 6 left Computer 3
Student 7 left Computer 2
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$
```

Exercise 4:



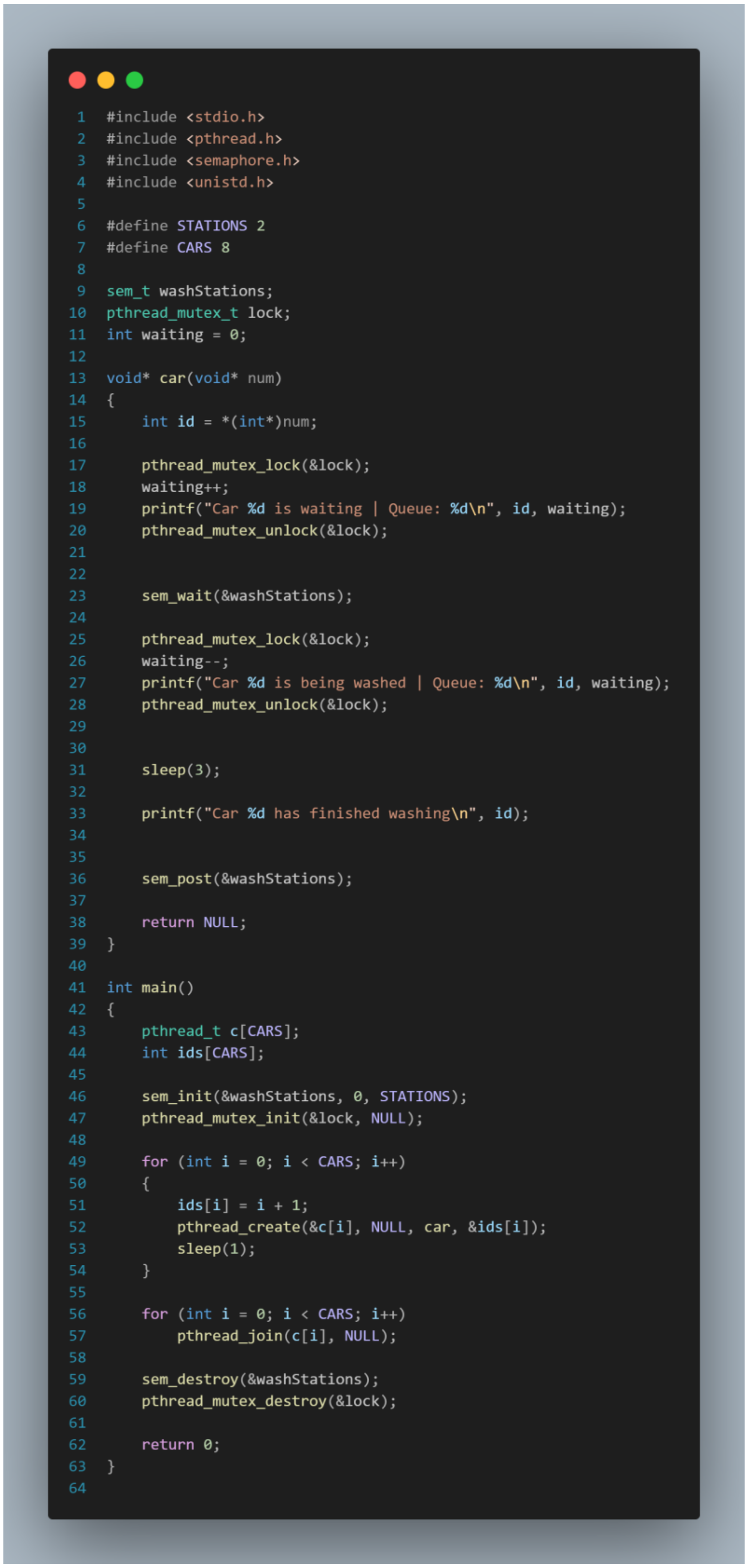
```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdlib.h>

#define WORKERS 3
#define TASKS 10

sem_t workerPool;

void* task(void* num)
{
    int id = *(int*)num;


    sem_wait(&workerPool);

    printf("Task %d is being processed by a worker\n", id);


    sleep(rand() % 2 + 1);

    printf("Task %d is finished\n", id);


    sem_post(&workerPool);

    return NULL;
}

int main()
{
    pthread_t t[TASKS];
    int ids[TASKS];


    sem_init(&workerPool, 0, WORKERS);

    for (int i = 0; i < TASKS; i++)
    {
        ids[i] = i + 1;
        pthread_create(&t[i], NULL, task, &ids[i]);
        sleep(1);
    }

    for (int i = 0; i < TASKS; i++)
    {
        pthread_join(t[i], NULL);
    }

    sem_destroy(&workerPool);

    return 0;
}
```

Output:



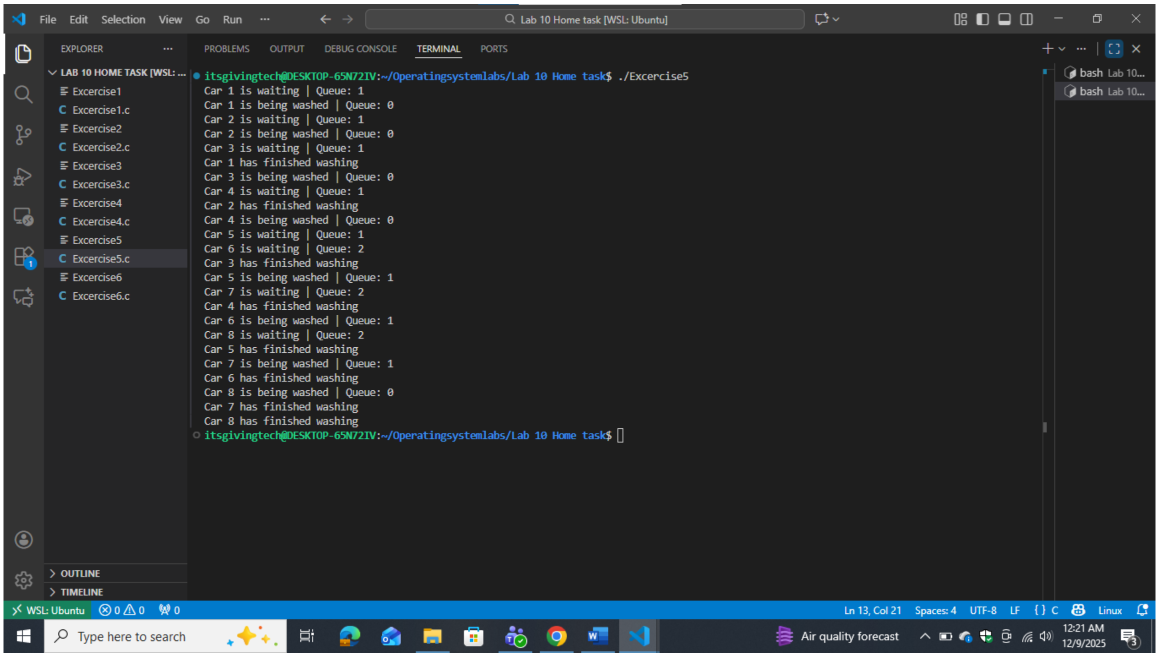
```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ gcc Excercise4.c -o Excercise4
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise4
Task 1 is being processed by a worker
Task 2 is being processed by a worker
Task 1 is finished
Task 2 is finished
Task 3 is being processed by a worker
Task 4 is being processed by a worker
Task 3 is finished
Task 5 is being processed by a worker
Task 4 is finished
Task 6 is being processed by a worker
Task 5 is finished
Task 7 is being processed by a worker
Task 6 is finished
Task 7 is finished
Task 8 is being processed by a worker
Task 8 is finished
Task 9 is being processed by a worker
Task 10 is being processed by a worker
Task 9 is finished
Task 10 is finished
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$
```

Exercise 5:



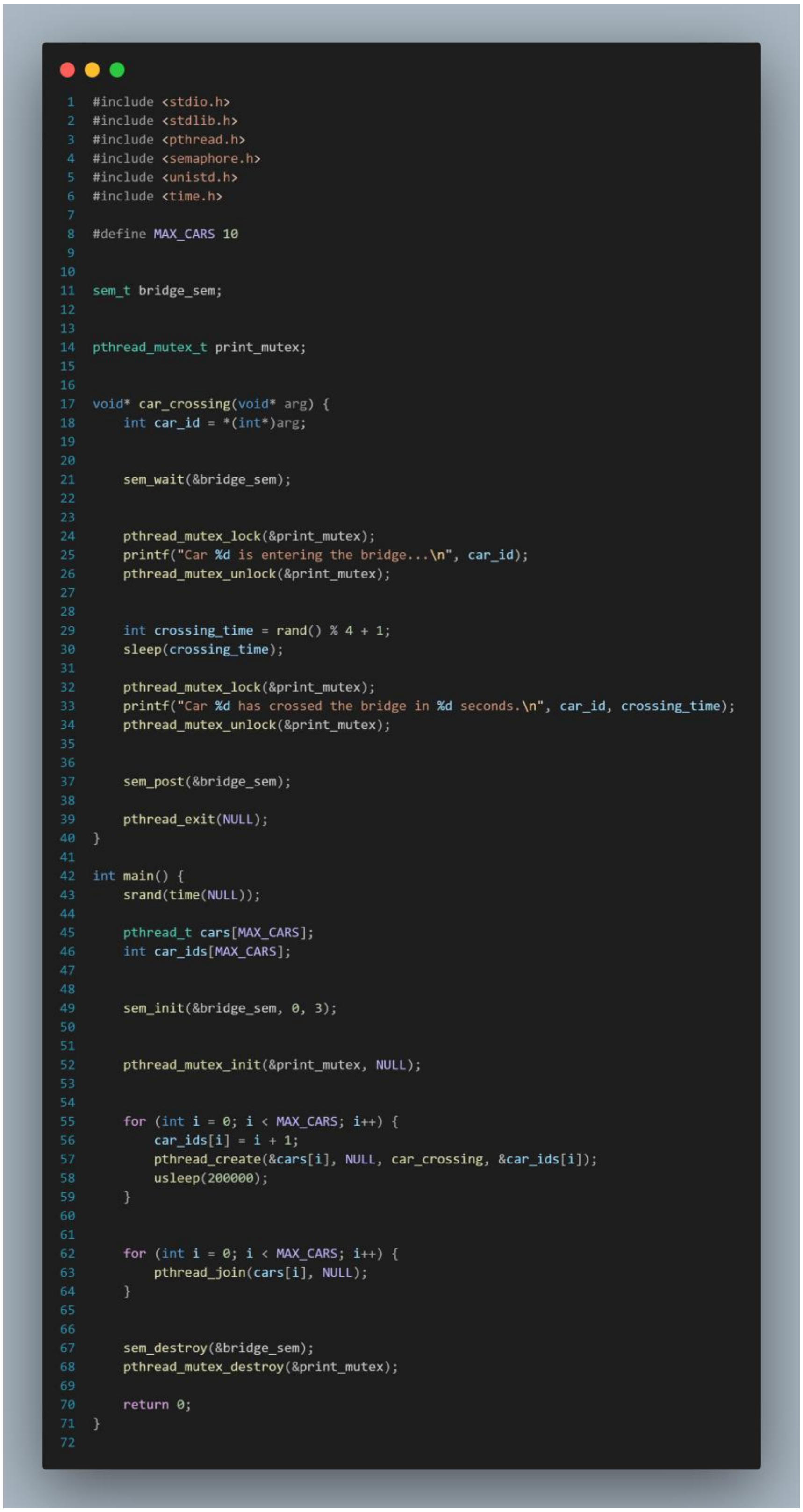
```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define STATIONS 2
#define CARS 8

sem_t washStations;
pthread_mutex_t lock;
int waiting = 0;

void* car(void* num)
{
    int id = *(int*)num;

    pthread_mutex_lock(&lock);
    waiting++;
    printf("Car %d is waiting | Queue: %d\n", id, waiting);
    pthread_mutex_unlock(&lock);


    sem_wait(&washStations);

    pthread_mutex_lock(&lock);
    waiting--;
    printf("Car %d is being washed | Queue: %d\n", id, waiting);
    pthread_mutex_unlock(&lock);


    sleep(3);

    printf("Car %d has finished washing\n", id);


    sem_post(&washStations);

    return NULL;
}

int main()
{
    pthread_t c[CARS];
    int ids[CARS];

    sem_init(&washStations, 0, STATIONS);
    pthread_mutex_init(&lock, NULL);

    for (int i = 0; i < CARS; i++)
    {
        ids[i] = i + 1;
        pthread_create(&c[i], NULL, car, &ids[i]);
        sleep(1);
    }

    for (int i = 0; i < CARS; i++)
        pthread_join(c[i], NULL);

    sem_destroy(&washStations);
    pthread_mutex_destroy(&lock);

    return 0;
}
```

Output:



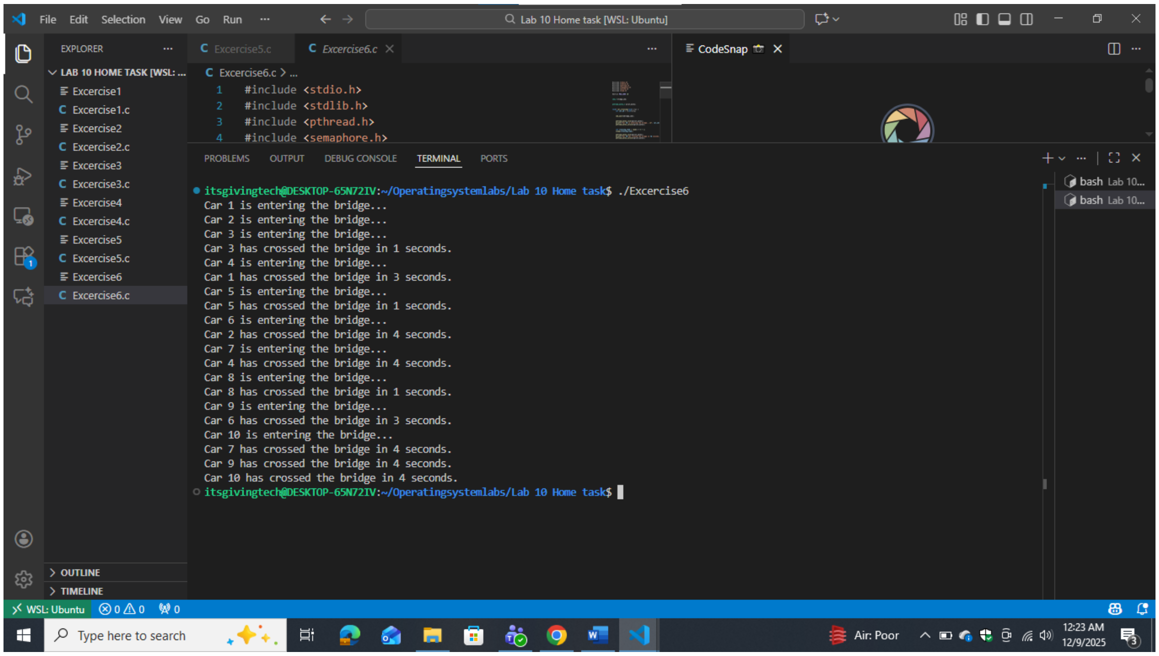
```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise5
Car 1 is waiting | Queue: 1
Car 1 is being washed | Queue: 0
Car 2 is waiting | Queue: 1
Car 2 is being washed | Queue: 0
Car 3 is waiting | Queue: 1
Car 1 has finished washing
Car 3 is being washed | Queue: 0
Car 4 is waiting | Queue: 1
Car 2 has finished washing
Car 4 is being washed | Queue: 0
Car 5 is waiting | Queue: 1
Car 6 is waiting | Queue: 2
Car 3 has finished washing
Car 5 is being washed | Queue: 1
Car 7 is waiting | Queue: 2
Car 4 has finished washing
Car 6 is being washed | Queue: 1
Car 8 is waiting | Queue: 2
Car 5 has finished washing
Car 7 is being washed | Queue: 1
Car 6 has finished washing
Car 8 is being washed | Queue: 0
Car 7 has finished washing
Car 8 has finished washing
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$
```

Exercise 6:



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

#define MAX_CARS 10


sem_t bridge_sem;


pthread_mutex_t print_mutex;


void* car_crossing(void* arg) {
    int car_id = *(int*)arg;


    sem_wait(&bridge_sem);


    pthread_mutex_lock(&print_mutex);
    printf("Car %d is entering the bridge...\n", car_id);
    pthread_mutex_unlock(&print_mutex);


    int crossing_time = rand() % 4 + 1;
    sleep(crossing_time);

    pthread_mutex_lock(&print_mutex);
    printf("Car %d has crossed the bridge in %d seconds.\n", car_id, crossing_time);
    pthread_mutex_unlock(&print_mutex);


    sem_post(&bridge_sem);

    pthread_exit(NULL);
}

int main() {
    srand(time(NULL));

    pthread_t cars[MAX_CARS];
    int car_ids[MAX_CARS];


    sem_init(&bridge_sem, 0, 3);


    pthread_mutex_init(&print_mutex, NULL);


    for (int i = 0; i < MAX_CARS; i++) {
        car_ids[i] = i + 1;
        pthread_create(&cars[i], NULL, car_crossing, &car_ids[i]);
        usleep(200000);
    }


    for (int i = 0; i < MAX_CARS; i++) {
        pthread_join(cars[i], NULL);
    }


    sem_destroy(&bridge_sem);
    pthread_mutex_destroy(&print_mutex);

    return 0;
}

```

Output:



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

```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise6
Car 1 is entering the bridge...
Car 2 is entering the bridge...
Car 3 is entering the bridge...
Car 3 has crossed the bridge in 1 seconds.
Car 4 is entering the bridge...
Car 1 has crossed the bridge in 3 seconds.
Car 5 is entering the bridge...
Car 5 has crossed the bridge in 1 seconds.
Car 6 is entering the bridge...
Car 2 has crossed the bridge in 4 seconds.
Car 7 is entering the bridge...
Car 4 has crossed the bridge in 4 seconds.
Car 8 is entering the bridge...
Car 8 has crossed the bridge in 1 seconds.
Car 9 is entering the bridge...
Car 6 has crossed the bridge in 3 seconds.
Car 10 is entering the bridge...
Car 7 has crossed the bridge in 4 seconds.
Car 9 has crossed the bridge in 4 seconds.
Car 10 has crossed the bridge in 4 seconds.
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$
```