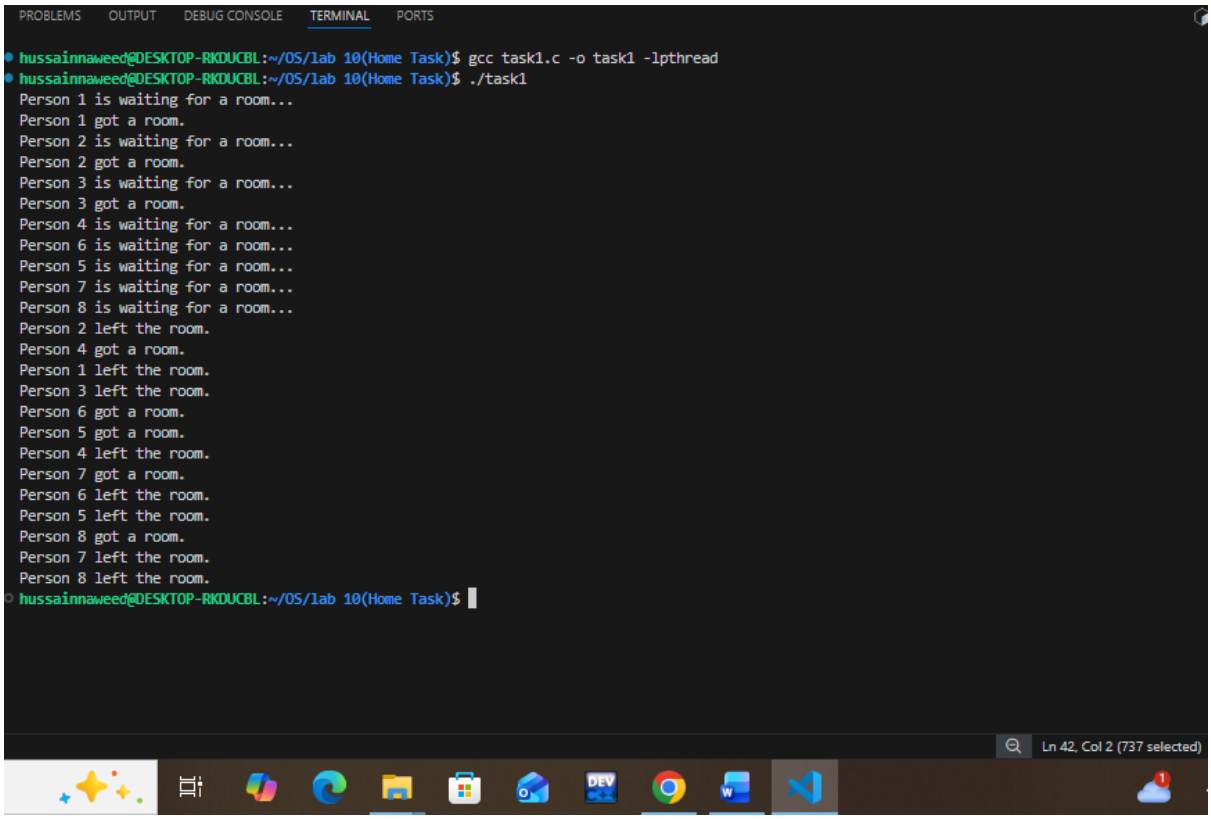
## Task1:

## Code:

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

#define ROOMS 3
#define PEOPLE 8

sem_t rooms;

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

   printf("Person %d is waiting for a room...\n", id);
   sem_wait(&rooms);

   printf("Person %d got a room.\n", id);
   sleep(2);

   printf("Person %d left the room.\n", id);
   sem_post(&rooms);

   return NULL;
}

int main() {
```

```
    pthread_t p[PEOPLE];
    int ids[PEOPLE];

    sem_init(&rooms, 0, ROOMS);

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

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

    sem_destroy(&rooms);
    return 0;
}
```

**Output:**




**Task 2:**

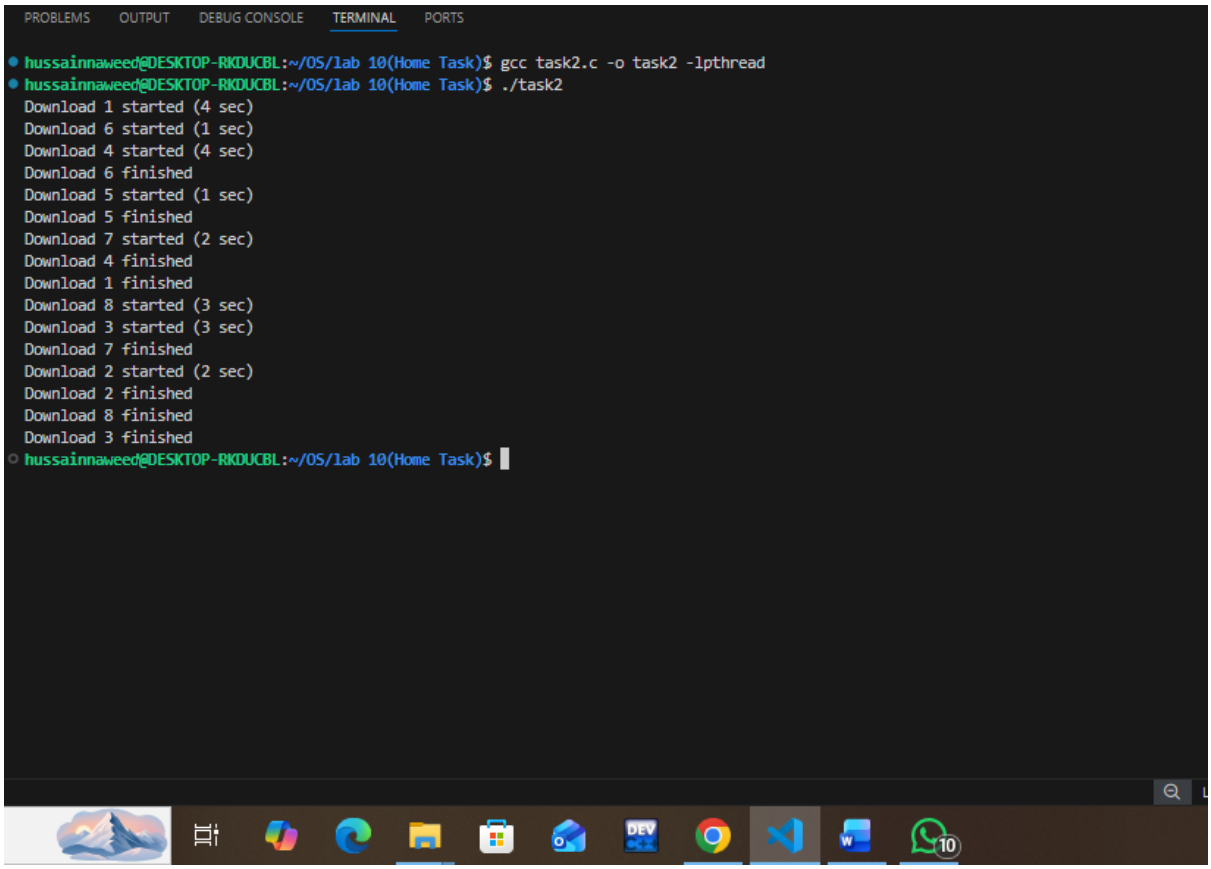
**Code:**

```
#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 download_slots;

void* download(void* arg) {
    int id = *(int*)arg;
    int time = rand() % 5 + 1;

    sem_wait(&download_slots);
    printf("Download %d started (%d sec)\n", id, time);

    sleep(time);

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

    return NULL;
}

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

    sem_init(&download_slots, 0, MAX_DOWNLOADS);
```

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

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

    sem_destroy(&download_slots);
    return 0;
}
```

**Output:**




## Task 3:

## Code:

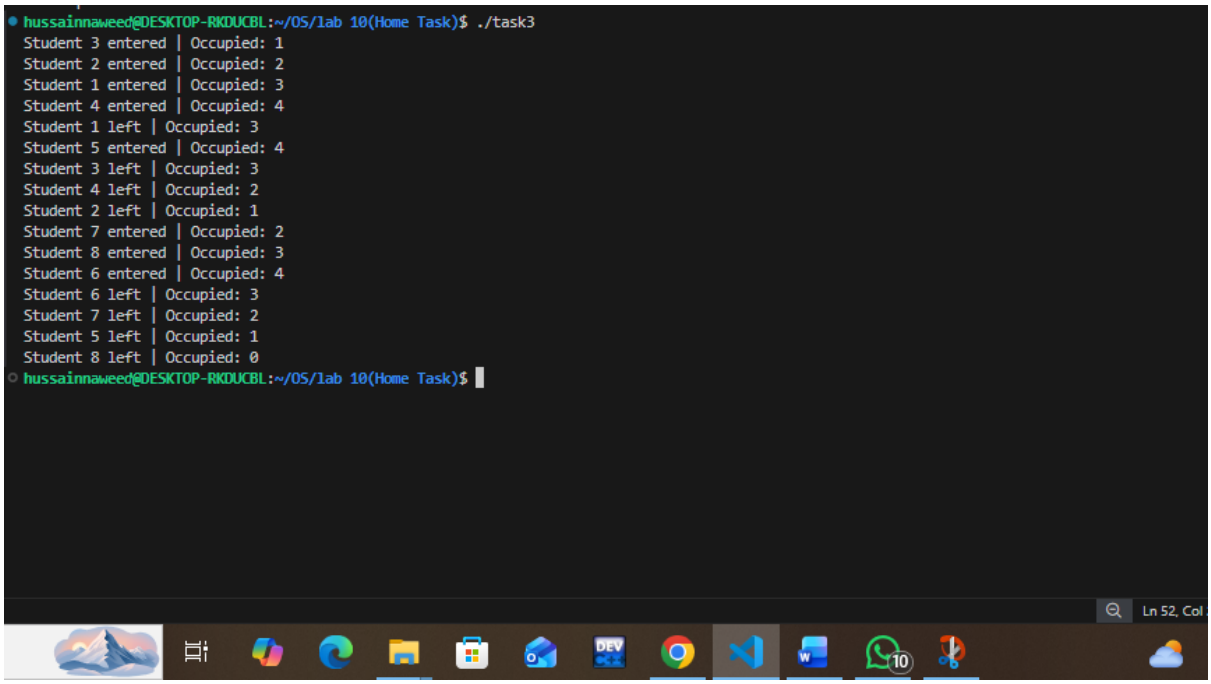
```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define COMPUTERS 4
#define STUDENTS 8
```

```
sem_t computers;
pthread_mutex_t mutex;
int occupied = 0;

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

    sem_wait(&computers);

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

    sleep(rand() % 3 + 1);

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

    sem_post(&computers);
```

```
    return NULL;
}

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

    sem_init(&computers, 0, COMPUTERS);
    pthread_mutex_init(&mutex, NULL);

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

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

    sem_destroy(&computers);
    pthread_mutex_destroy(&mutex);
    return 0;
```

}

## Output:




## Task 4:

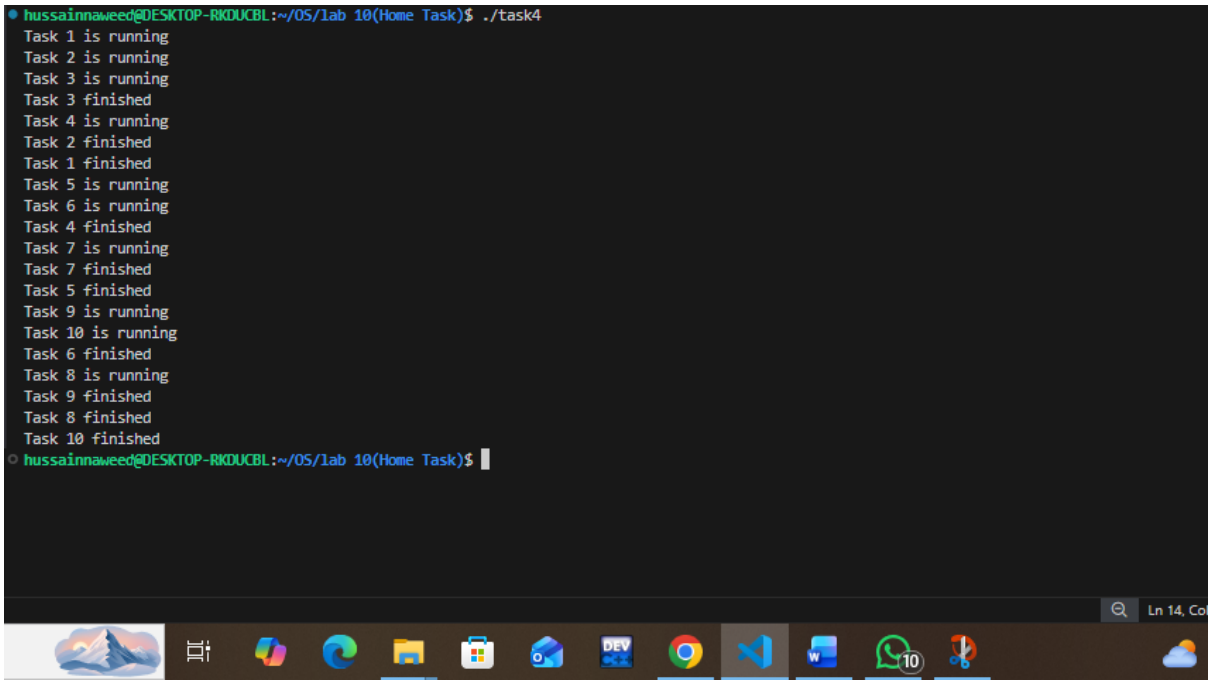
## Code:

```
#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


#define WORKERS 3

#define TASKS 10


sem_t workers;
```

```
void* task(void* arg) {
    int id = *(int*)arg;

    sem_wait(&workers);
    printf("Task %d is running\n", id);

    sleep(rand() % 2 + 1);

    printf("Task %d finished\n", id);
    sem_post(&workers);

    return NULL;
}

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

    sem_init(&workers, 0, WORKERS);

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

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

```
    sem_destroy(&workers);
    return 0;
}
```

## Output:




## Task 5:

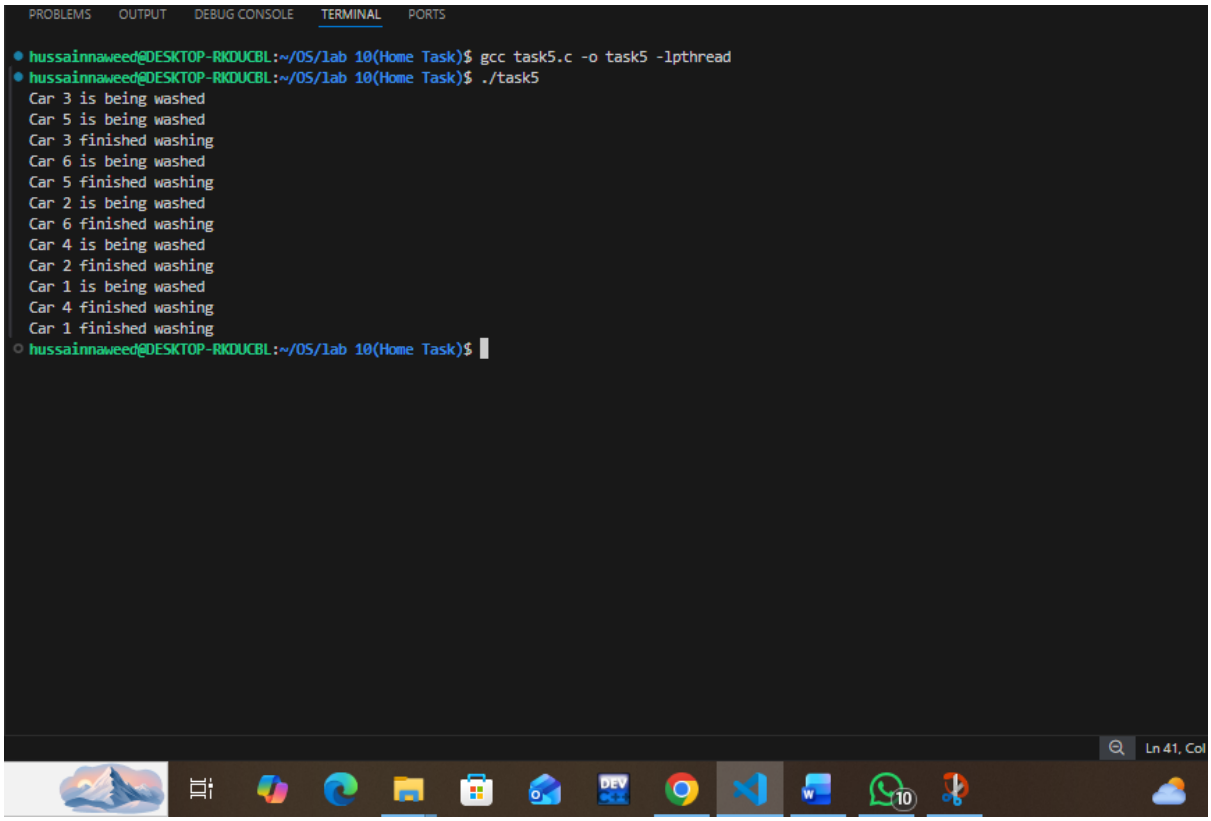
## Code:

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

#define STATIONS 2
#define CARS 6

sem_t wash;
```

```
void* car(void* arg) {
    int id = *(int*)arg;

    sem_wait(&wash);
    printf("Car %d is being washed\n", id);

    sleep(3);

    printf("Car %d finished washing\n", id);
    sem_post(&wash);

    return NULL;
}

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

    sem_init(&wash, 0, STATIONS);

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

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

```
    sem_destroy(&wash);
    return 0;
}
```

## Output:




## Task 6:

## Code:

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

        #define MAX_CARS 3
        #define TOTAL_CARS 8
```

```
sem_t bridge;
pthread_mutex_t print;

void* car(void* arg) {
    int id = *(int*)arg;
    int time = rand() % 3 + 1;

    sem_wait(&bridge);

    pthread_mutex_lock(&print);
    printf("Car %d entered bridge\n", id);
    pthread_mutex_unlock(&print);

    sleep(time);

    pthread_mutex_lock(&print);
    printf("Car %d left bridge\n", id);
    pthread_mutex_unlock(&print);

    sem_post(&bridge);
    return NULL;
}

int main() {
    pthread_t c[TOTAL_CARS];
    int ids[TOTAL_CARS];
```

```
    sem_init(&bridge, 0, MAX_CARS);
    pthread_mutex_init(&print, NULL);

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

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

    sem_destroy(&bridge);
    pthread_mutex_destroy(&print);
    return 0;
}
```

## Output: