Gebze Technical University
Computer Engineering Department

CSE 344 – System Programming

Homework-3 Report

Muhammet Akkurt

1901042644

Purpose and Details of the Assignment:

In this parking system, it simulates receiving the vehicle of the incoming customer and parking it in the temporary parking lot and transferring the filled vehicle type to the permanent parking lot when the temporary parking lot is full. The system manages vehicle arrivals, parking allocations, and transitions between temporary and permanent parking spaces using multithreading and synchronization mechanisms.

Vehicles arriving at the system are picked up from the customer by two valets (one for each vehicle type) and parked in the temporary parking lot. If a vehicle type in the temporary parking lot reaches its maximum, it is transferred there by the valet if there is a space in the permanent parking lot. Thus, the temporary parking space for the vehicle type is reset and the customer continues to be accepted. When the temporary parking space is full for a vehicle, if there is no space in the permanent parking lot, the current customer and the customers of the same vehicle type after him/her are turned away from the parking lot.

Let's examine the synchronization and operation logic in more detail on the code:

The constants define the capacity of the parking lot, both temporary and permanent.

These defined semaphores are used to signal the arrival of new vehicles in the parking lot (newAutomobile and newPickup), to coordinate parking operations between car owners and attendants (inChargeforAutomobile and inChargeforPickup) and to signal program termination (exitSignal).

The system uses integer counters (mFree_automobile, mFree_pickup, pFree_automobile, pFree_pickup) to monitor the number of free parking spaces in both temporary and permanent parking lots for cars and pickups.

To protect these counters from simultaneous access by multiple threads, which could lead to inconsistent or corrupt data, mutexes are employed. There are separate mutexes for the counters of permanent spots (automobile_mutex and pickup_mutex) and temporary spots (temp_automobile_mutex and temp_pickup_mutex). These mutexes ensure that only one thread can modify the counters at a time, thus avoiding race conditions and ensuring the integrity of the data.

```c
#define PERMA_AUTOMOBILE_SPACES 8
#define PERMA_PICKUP_SPACES 4
#define TEMP_AUTOMOBILE_SPACES 8
#define TEMP_PICKUP_SPACES 4

/*Semaphore declarations*/
sem_t newAutomobile, newPickup;
sem_t inChargeforAutomobile, inChargeforPickup;
sem_t exitSignal;

/*Counter variables for free spots*/
int mFree_automobile = TEMP_AUTOMOBILE_SPACES;
int mFree_pickup = TEMP_PICKUP_SPACES;
int pFree_automobile = PERMA_AUTOMOBILE_SPACES;
int pFree_pickup = PERMA_PICKUP_SPACES;

/*Mutex for protecting the counters*/
pthread_mutex_t automobile_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t pickup_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t temp_automobile_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t temp_pickup_mutex = PTHREAD_MUTEX_INITIALIZER;
```

The initialization of semaphores, which manage the entry of vehicles and the coordination between vehicle owners and parking attendants for both cars and pickups, is done.

Threads are then created for the car owners simulating vehicle arrivals and for the attendants managing the parking lot. A total of three threads are created, one to simulate the car owners and two for the valets of two different car types. The main thread waits for a semaphore signal (exitSignal) indicating that the parking lot is full and triggers the program to start shutdown procedures. This involves canceling and merging all running threads to ensure that all resources are properly freed and no processes are left hanging. After thread management, semaphores are destroyed and synchronization tools are effectively cleaned up to prevent resource leaks.

```c
int main() {
    srand(time(NULL));
    char buffer[256];
    int len;
    pthread_t owner_thread, automobile_attendant_thread, pickup_attendant_thread;
    int automobile = 0, pickup = 1;

    /*Initialize semaphores*/
    sem_init(&newAutomobile, 0, 0);
    sem_init(&newPickup, 0, 0);
    sem_init(&inChargeforAutomobile, 0, 0);
    sem_init(&inChargeforPickup, 0, 0);
    sem_init(&exitSignal, 0, 0);

    /*Create threads*/
    pthread_create(&owner_thread, NULL, carOwner, NULL);
    pthread_create(&automobile_attendant_thread, NULL, carAttendant, &automobile);
    pthread_create(&pickup_attendant_thread, NULL, carAttendant, &pickup);

    /*Wait for the exit signal*/
    sem_wait(&exitSignal);

    /*Cancel and join threads before exiting*/
    pthread_cancel(owner_thread);
    pthread_join(owner_thread, NULL);
    pthread_cancel(automobile_attendant_thread);
    pthread_join(automobile_attendant_thread, NULL);
    pthread_cancel(pickup_attendant_thread);
    pthread_join(pickup_attendant_thread, NULL);

    /* Destroy semaphores*/
    sem_destroy(&newAutomobile);
    sem_destroy(&newPickup);
    sem_destroy(&inChargeforAutomobile);
    sem_destroy(&inChargeforPickup);
    sem_destroy(&exitSignal);

    /*Log that all parking spaces are full and the program is exiting*/
    len = snprintf(buffer, sizeof(buffer), "All parking spaces are full. Exiting the program.\n");
    write(STDOUT_FILENO, buffer, len);

    return 0;
}
```

This function continuously simulates the arrival of vehicles to a parking lot. In the function, a loop runs indefinitely, and in each iteration, a vehicle type is randomly determined-either an automobile (0) or a pickup (1). Between each iteration, the function pauses for a random period (up to 5 seconds), simulating the variable arrival times of vehicles. The function then attempts to park the vehicle using a mutex lock to ensure exclusive access to the parking spot counters for each vehicle type. If a spot is available(checked by the counters mFree_automobile or mFree_pickup), it decrements the counter to indicate that a spot has been taken and logs a message to the standard output detailing the arrival and successful parking of the vehicle. The function then signals another thread (automobile or pickup attendant) using semaphores (newAutomobile or newPickup) to manage the vehicle's parking process further and waits for an attendant thread to complete the parking via “inChargeforAutomobile” or “inChargeforPickup”.

If no temporary parking space is available, the mutex is unlocked without decrementing the space counter, and a message is logged indicating that the vehicle has arrived but is leaving due to the lack of available space.

```c
void* carOwner(void* arg) {         /*Thread function for car owners*/
    char buffer[256];
    int len;
    while (1) {
        int vehicle_type = rand() % 2;  /*0 for automobile, 1 for pickup*/

        if (vehicle_type == 0) {  /*If automobile*/
            pthread_mutex_lock(&temp_automobile_mutex);

            if (mFree_automobile > 0) {
                mFree_automobile--;
                len = snprintf(buffer, sizeof(buffer), "The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: %d\n", mFree_automobile+1);
                write(STDOUT_FILENO, buffer, len);
                pthread_mutex_unlock(&temp_automobile_mutex);
                sem_post(&newAutomobile);              /*Signal the arrival of a new automobile*/
                sem_wait(&inChargeforAutomobile);      /*Wait for the attendant to park the car*/
            } else {
                pthread_mutex_unlock(&temp_automobile_mutex);
                len = snprintf(buffer, sizeof(buffer), "The automobile arrives at the parking lot entrance but there is no temporary space for automobiles. The automobile owner is leaving.\n\n");
                write(STDOUT_FILENO, buffer, len);
            }
        } else {                    /*If pickup*/
            pthread_mutex_lock(&temp_pickup_mutex);
            if (mFree_pickup > 0) {
                mFree_pickup--;
                len = snprintf(buffer, sizeof(buffer), "The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: %d\n", mFree_pickup+1);
                write(STDOUT_FILENO, buffer, len);
                pthread_mutex_unlock(&temp_pickup_mutex);
                sem_post(&newPickup);           /* Signal the arrival of a new pickup*/
                sem_wait(&inChargeforPickup);   /*Wait for the attendant to park the pickup*/
            } else {
                pthread_mutex_unlock(&temp_pickup_mutex);
                len = snprintf(buffer, sizeof(buffer), "The pickup arrives at the parking lot entrance but there is no temporary space for pickups. The pickup owner is leaving.\n\n\n");
                write(STDOUT_FILENO, buffer, len);
            }
        }
        sleep(rand() % 5);          /*Simulate the time before the next car arrives*/
    }
}
```

This function operates continuously within its own thread, designated for either automobiles or pickups based on the vehicle_type parameter passed to it. Upon a vehicle's arrival, signaled by the newAutomobile or newPickup semaphore, the function simulates the parking process by waiting (sleeping for a second) to mimic the time it takes to park a vehicle.

After parking a vehicle in a temporary spot, the function logs the action to the standard output and signals back to the vehicle owner thread that the parking process has completed using inChargeforAutomobile or inChargeforPickup semaphore. It uses mutex locks (temp_automobile_mutex or temp_pickup_mutex) to ensure that access to the temporary parking counters is synchronized to prevent race conditions.

When the temporary parking for a vehicle type is full, the attendant checks if there is space in the permanent parking area. If space is available, the function transitions vehicles from the temporary to the permanent parking spaces. This is managed under multiple mutex locks (automobile_mutex, pickup_mutex, temp_automobile_mutex, and temp_pickup_mutex) to synchronize access to both temporary and permanent counters, ensuring the integrity of operations across different vehicle types and parking areas. It updates the counters and logs each movement of a vehicle from temporary to permanent parking.

Throughout its operation, the function continuously logs the status of the parking space after each vehicle is parked or moved.

Finally, the function checks if all parking spaces (both temporary and permanent for all vehicle types) are full. If they are full, it sends a message to the exitSignal semaphore, indicating to the main thread that it can terminate the program because the parking space is now full.

```c
void* carAttendant(void* arg) {            /*Thread function for car attendants*/
    int vehicle_type = *(int*)arg;
    char buffer[256];
    int len;
    while (1) {
        if (vehicle_type == 0) {                        /*Attendant for automobiles*/
            sem_wait(&newAutomobile);                       /*Wait for a new automobile to arrive*/
            sleep(1);                                   /*Simulate parking time in temporary space*/
            pthread_mutex_lock(&temp_automobile_mutex);
            len = snprintf(buffer, sizeof(buffer), "The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: %d\n", mFree_automobile);
            write(STDOUT_FILENO, buffer, len);
            sem_post(&inChargeforAutomobile);                   /*Signal that the car has been parked*/
            pthread_mutex_unlock(&temp_automobile_mutex);

            if (mFree_automobile == 0) {                /*Temporary parking full, move to permanent*/
                pthread_mutex_lock(&automobile_mutex);
                pthread_mutex_lock(&pickup_mutex);
                pthread_mutex_lock(&temp_automobile_mutex);
                pthread_mutex_lock(&temp_pickup_mutex);
                len = snprintf(buffer, sizeof(buffer), "\nThe temporary automobile parking lot is full. If there is space in the permanent park, the valet will start moving the automobiles.\n");
                write(STDOUT_FILENO, buffer, len);
                while (pFree_automobile > 0 && mFree_automobile < TEMP_AUTOMOBILE_SPACES) {
                    mFree_automobile++;
                    pFree_automobile--;
                    len = snprintf(buffer, sizeof(buffer), "Automobile permanently parked. Available permanent automobile parking spaces: %d, Available temporary automobile parking spaces: %d\n",
                            pFree_automobile, mFree_automobile);
                    write(STDOUT_FILENO, buffer, len);
                    sleep(1);                           /*Simulate moving time*/
                }
                pthread_mutex_unlock(&temp_pickup_mutex);
                pthread_mutex_unlock(&temp_automobile_mutex);
                pthread_mutex_unlock(&pickup_mutex);
                pthread_mutex_unlock(&automobile_mutex);
            }
        } else {                                /*Attendant for pickups*/
            sem_wait(&newPickup);               /*Wait for a new pickup to arrive*/
            sleep(1);                           /*Simulate parking time in temporary space*/
            pthread_mutex_lock(&temp_pickup_mutex);
            len = snprintf(buffer, sizeof(buffer), "The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: %d\n", mFree_pickup);
            write(STDOUT_FILENO, buffer, len);
            sem_post(&inChargeforPickup);               /*Signal that the pickup has been parked*/
            pthread_mutex_unlock(&temp_pickup_mutex);

            if (mFree_pickup == 0) {                    /*Temporary parking full, move to permanent*/
                pthread_mutex_lock(&automobile_mutex);
                pthread_mutex_lock(&pickup_mutex);
                pthread_mutex_lock(&temp_automobile_mutex);
                pthread_mutex_lock(&temp_pickup_mutex);
                len = snprintf(buffer, sizeof(buffer), "\nThe temporary pickup parking lot is full. If there is space in the permanent park, the valet will start moving the pickups.\n");
                write(STDOUT_FILENO, buffer, len);
                while (pFree_pickup > 0 && mFree_pickup < TEMP_PICKUP_SPACES) {
                    mFree_pickup++;
                    pFree_pickup--;
                    len = snprintf(buffer, sizeof(buffer), "Pickup permanently parked. Available permanent pickup parking spaces: %d, Available temporary pickup parking spaces: %d\n",
                            pFree_pickup, mFree_pickup);
                    write(STDOUT_FILENO, buffer, len);
                    sleep(1);                               /*Simulate moving time*/
                }
                pthread_mutex_unlock(&temp_pickup_mutex);
                pthread_mutex_unlock(&temp_automobile_mutex);
                pthread_mutex_unlock(&pickup_mutex);
                pthread_mutex_unlock(&automobile_mutex);
            }
```

```c
logStatus();                            /*Log the status after each parking operation*/

pthread_mutex_lock(&automobile_mutex);
pthread_mutex_lock(&pickup_mutex);
pthread_mutex_lock(&temp_automobile_mutex);
pthread_mutex_lock(&temp_pickup_mutex);

if (mFree_automobile == 0 && pFree_automobile == 0 && mFree_pickup == 0 && pFree_pickup == 0) {     /*Check if all parking spots are full*/
    sem_post(&exitSignal);  /*Signal main thread to exit*/
}

pthread_mutex_unlock(&temp_pickup_mutex);
pthread_mutex_unlock(&temp_automobile_mutex);
pthread_mutex_unlock(&pickup_mutex);
pthread_mutex_unlock(&automobile_mutex);
```

The availability of temporary and permanent parking spaces for all vehicle types is logged after each parking. Mutex locks are used to prevent simultaneous access to the variables used in this process.

```c
void logStatus() {          /*Function to log the current status of parking spaces*/
    char buffer[256];
    int len;
    pthread_mutex_lock(&automobile_mutex);
    pthread_mutex_lock(&pickup_mutex);
    pthread_mutex_lock(&temp_automobile_mutex);
    pthread_mutex_lock(&temp_pickup_mutex);

    len = snprintf(buffer, sizeof(buffer), "Temporary Automobile Parking Spaces: %d/%d, Temporary Pickup Parking Spaces: %d/%d\n",
                   TEMP_AUTOMOBILE_SPACES - mFree_automobile, TEMP_AUTOMOBILE_SPACES,
                   TEMP_PICKUP_SPACES - mFree_pickup, TEMP_PICKUP_SPACES);
    write(STDOUT_FILENO, buffer, len);

    len = snprintf(buffer, sizeof(buffer), "Permanent Automobile Parking Spaces: %d/%d, Permanent Pickup Parking Spaces: %d/%d\n\n\n",
                   PERMA_AUTOMOBILE_SPACES - pFree_automobile, PERMA_AUTOMOBILE_SPACES,
                   PERMA_PICKUP_SPACES - pFree_pickup, PERMA_PICKUP_SPACES);
    write(STDOUT_FILENO, buffer, len);

    pthread_mutex_unlock(&temp_pickup_mutex);
    pthread_mutex_unlock(&temp_automobile_mutex);
    pthread_mutex_unlock(&pickup_mutex);
    pthread_mutex_unlock(&automobile_mutex);
}
```

Test Scenarios:

The program is started and randomly generated vehicles start arriving at the parking lot entrance. With the mutex locks used, only 1 vehicle is allowed to enter.

```
ubuntu@DESKTOP-A76HGBA:/mnt/c/Users/makkr/OneDrive/Masaüstü/Yeni klasör/cse344/hw3$ make
gcc -pthread main.c -o ParkingSystem
ubuntu@DESKTOP-A76HGBA:/mnt/c/Users/makkr/OneDrive/Masaüstü/Yeni klasör/cse344/hw3$ ./ParkingSystem
The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 4
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 3
Temporary Automobile Parking Spaces: 0/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 0/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 8
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 7
Temporary Automobile Parking Spaces: 1/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 0/4


The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 3
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 2
Temporary Automobile Parking Spaces: 1/8, Temporary Pickup Parking Spaces: 2/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 0/4


The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 2
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 1
Temporary Automobile Parking Spaces: 1/8, Temporary Pickup Parking Spaces: 3/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 0/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 7
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 6
Temporary Automobile Parking Spaces: 2/8, Temporary Pickup Parking Spaces: 3/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 0/4
```

When one of the vehicle types is full for the temporary parking lot, the valet moves the parked vehicles to the permanent parking lot. It is ensured that no new vehicles are taken during the transportation process.

```
The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 1
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 0

The temporary pickup parking lot is full. If there is space in the permanent park, the valet will start moving the pickups.
Pickup permanently parked. Available permanent pickup parking spaces: 3, Available temporary pickup parking spaces: 1
Pickup permanently parked. Available permanent pickup parking spaces: 2, Available temporary pickup parking spaces: 2
Pickup permanently parked. Available permanent pickup parking spaces: 1, Available temporary pickup parking spaces: 3
Pickup permanently parked. Available permanent pickup parking spaces: 0, Available temporary pickup parking spaces: 4
Temporary Automobile Parking Spaces: 2/8, Temporary Pickup Parking Spaces: 0/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 4/4


The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 4
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 3
Temporary Automobile Parking Spaces: 2/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 4/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 6
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 5
Temporary Automobile Parking Spaces: 3/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 4/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 5
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 4
Temporary Automobile Parking Spaces: 4/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 4/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 4
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 3
Temporary Automobile Parking Spaces: 5/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 4/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 3
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 2
Temporary Automobile Parking Spaces: 6/8, Temporary Pickup Parking Spaces: 1/4
Permanent Automobile Parking Spaces: 0/8, Permanent Pickup Parking Spaces: 4/4
```

When the temporary and permanent parking lots are completely full for a vehicle type, new customers of the same vehicle type are turned away.

```
The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 8
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 7
Temporary Automobile Parking Spaces: 1/8, Temporary Pickup Parking Spaces: 2/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4


The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 2
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 1
Temporary Automobile Parking Spaces: 1/8, Temporary Pickup Parking Spaces: 3/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4


The pickup arrives at the parking lot entrance. Available temporary pickup parking spaces: 1
The pickup was parked in a temporary parking lot by the valet. Available temporary pickup parking spaces: 0

The temporary pickup parking lot is full. If there is space in the permanent park, the valet will start moving the pickups.
Temporary Automobile Parking Spaces: 1/8, Temporary Pickup Parking Spaces: 4/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4


The pickup arrives at the parking lot entrance but there is no temporary space for pickups. The pickup owner is leaving.


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 7
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 6
Temporary Automobile Parking Spaces: 2/8, Temporary Pickup Parking Spaces: 4/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 6
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 5
Temporary Automobile Parking Spaces: 3/8, Temporary Pickup Parking Spaces: 4/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4
```

When the temporary and permanent parking spaces for all vehicle types are full, the system is terminated by sending a signal with a semaphore.

```
The pickup arrives at the parking lot entrance but there is no temporary space for pickups. The pickup owner is leaving.


The pickup arrives at the parking lot entrance but there is no temporary space for pickups. The pickup owner is leaving.


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 2
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 1
Temporary Automobile Parking Spaces: 7/8, Temporary Pickup Parking Spaces: 4/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4


The automobile arrives at the parking lot entrance. Available temporary automobile parking spaces: 1
The automobile was parked in a temporary parking lot by the valet. Available temporary automobile parking spaces: 0

The temporary automobile parking lot is full. If there is space in the permanent park, the valet will start moving the automobiles.
Temporary Automobile Parking Spaces: 8/8, Temporary Pickup Parking Spaces: 4/4
Permanent Automobile Parking Spaces: 8/8, Permanent Pickup Parking Spaces: 4/4


All parking spaces are full. Exiting the program.
ubuntu@DESKTOP-A76HGBA:/mnt/c/Users/makkr/OneDrive/Masaüstü/Yeni klasör/cse344/hw3$
```