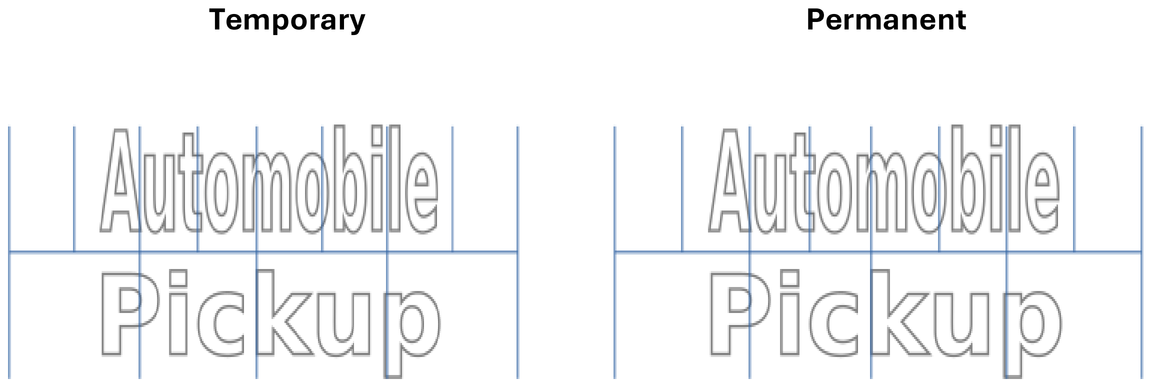
CSE 344-HW3 REPORT

In my program I have 2 parking lots, one is temporary, and one is permanent.
Each has 8 spaces for automobiles and 4 spaces for pickups.



I keep track of empty spaces with these integers which are a part of shared memory **ParkingLot** struct:

- **mFree_automobile_temp** : available spots in temp parking lot for automobiles
- **mFree_pickup_temp** : available spots in temporary parking lot for pickups
- **mFree_automobile** : available spots in permanent parking lot for automobiles
- **mFree_pickup** : available spots in permanent parking lot for pickups

New automobile acceptance criteria:

If **mFree_automobile_temp** + **mFree_automobile** > **MAX_AUTO_AMOUNT (8)**
This means that all automobiles in temp space should be able to be parked in permanent space without a spacing problem. Cars in temp + cars in permanent cannot be bigger than max automobile spots in permanent parking lot.
Also **mFree_automobile_temp** and **mFree_automobile** should not be 0.

New pickup acceptance criteria:

If **mFree_pickup_temp** + **mFree_pickup** > **MAX_PICKUP_AMOUNT (4)**
Same as automobile.

The system has 2 valet threads and 1 owner thread:

- **ownerThread**: This thread simulates car arrivals. Executes **carOwner** function.
- **attendantAutoThread**: Valet for automobiles. Executes **carAttendantAuto** function.

attendantPickupThread: Valet for pickups. Executes **carAttendantPickup** function.

The system has 2 automobile semaphores and 2 pickup semaphores:

newAutomobile: This semaphore is initialized as 0 and whenever a new automobile arrives, the semaphore is incremented by calling sem_post() function. This indicates a new car is waiting in the temporary parking space. So, automobile valet can take the car when he is available. Before valet starts parking the car, the semaphore is decremented by calling sem_wait() function which indicates that one of the cars in temporary parking space is no longer waiting to be parked.

inChargeforAutomobile: This semaphore is initialized as 1 and indicates that automobile valet is available. Valet will only be available to park a car when this semaphore is 1. carAttendantAuto function calls sem_wait() and waits for this semaphore to become 1 if it's 0 or decreases this semaphore to 0 when it is 1 and starts parking. After parking the car is done sem_post() function is called, and semaphore is re-incremented to 1. This ensures that only 1 car of this type will be parked at the same time.

newPickup: Same with newAutomobile.
inChargeforPickup: Same with inChargeforAutomobile

The system has 1 mutex:

parkingLotMutex: This mutex is used to synchronize access to the parkingLot struct, which is shared memory between multiple threads. I lock the mutex before reading from shared memory or changing a value in shared memory and unlock it afterwards.

How does the program work?

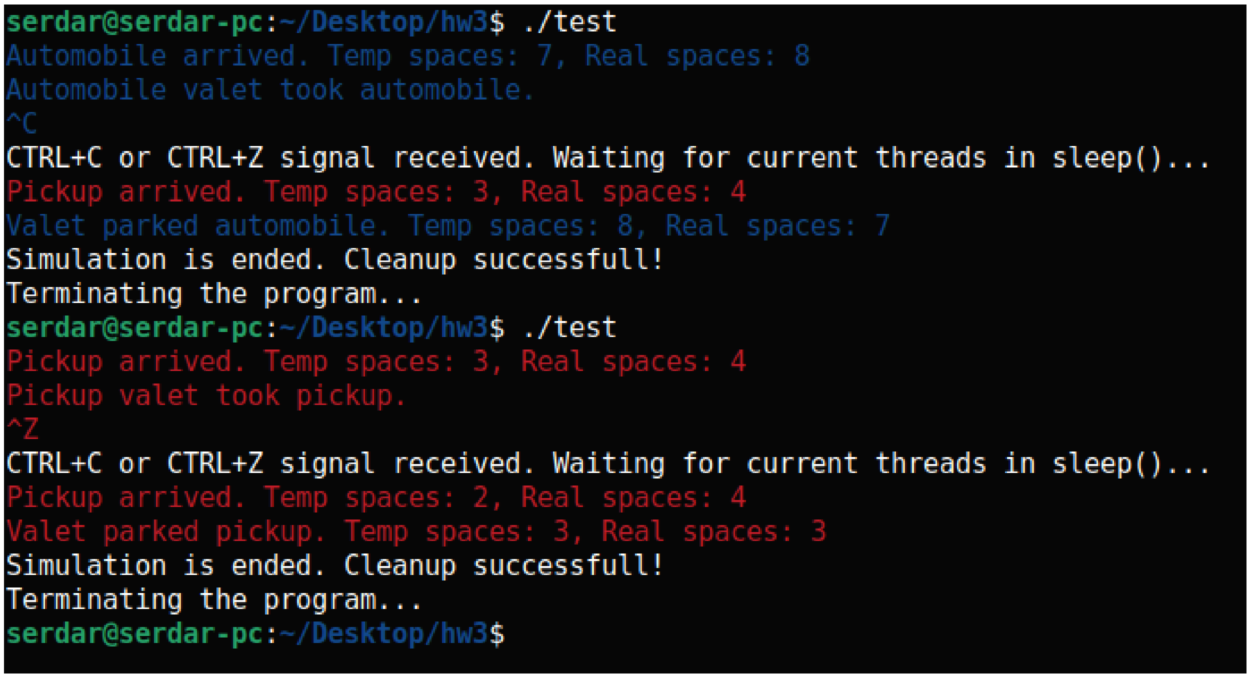
1) Both valets wait for new cars to arrive. In **carAttendantAuto** this is ensured with **sem_wait(&newAutomobile).** And In **carAttendantPickup** this is ensured with **sem_wait(&newPickup).**
2) A car is generated in **carOwner** function every **CAR_ARRIVAL_TIME** (default: 2 secs). This car is an automobile or a pickup by %50 chance. In our case let's assume that the first car is a **pickup**.
3) For the new car, acceptance criteria check will be done. If there is empty space in parking lots, **mFree_pickup_temp** will be decreased and **sem_post(&newPickup)** will be called which means a new car arrived and valet can park it when he is available. If there is no space in parking lots, then no sem_post will be called.

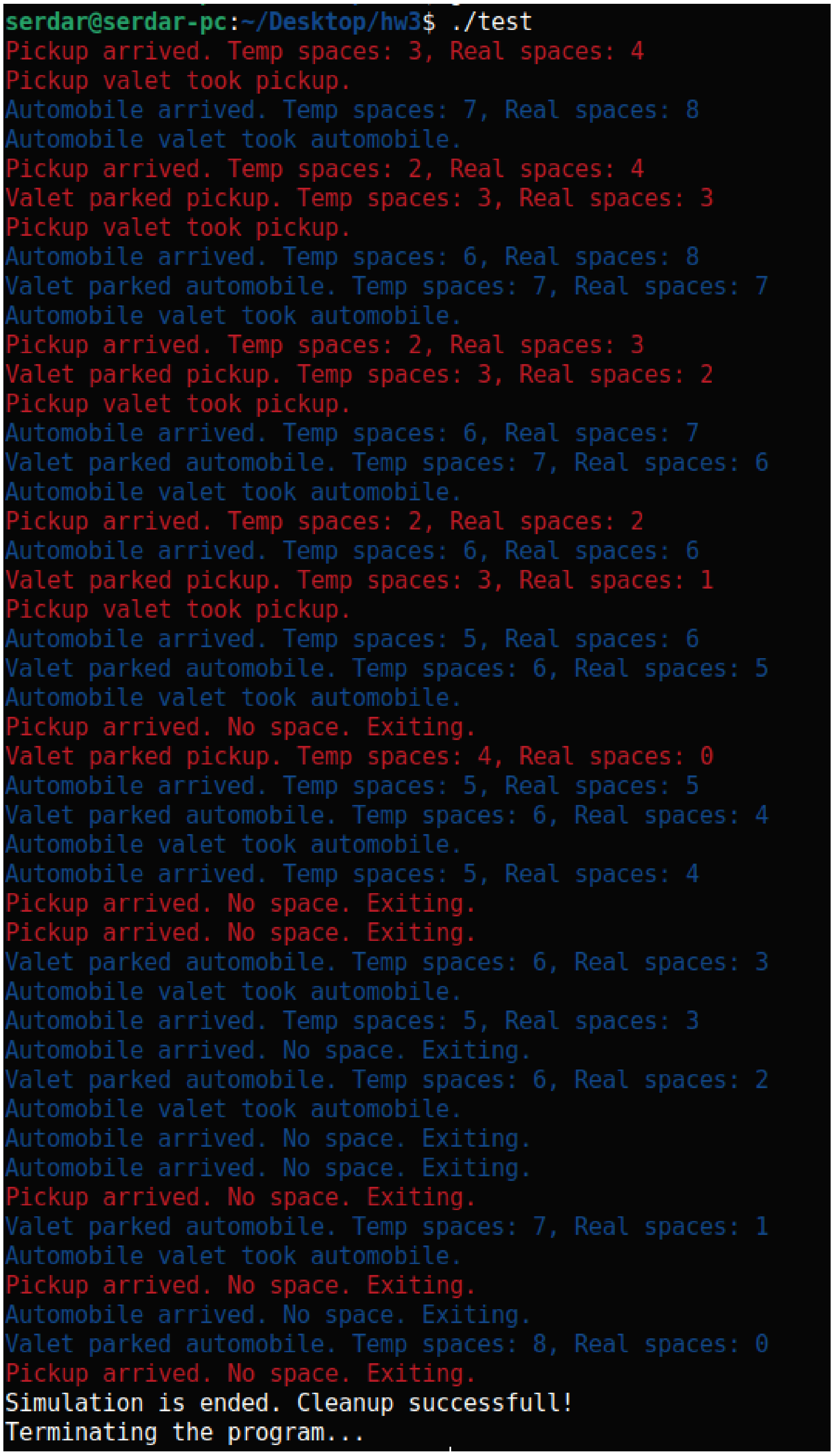
4) In **carAttendantPickup** the **sem_wait(&newPickup)** function will stop waiting because sem_post is called in another thread and newPickup semaphore became 1. With the help of this wait function the semaphore will be decreased to 0 again.
5) Now **sem_wait(&inChargeforPickup)** will be called and function will wait for valet to be available. When valet is available this semaphore will be decreased to 0. Parking will be simulated, which takes **VALET_PARK_TIME** (default: 5) seconds.
6) After parking, **mFree_pickup_temp** will be increased again because the pickup is no longer in temporary parking space but now **mFree_pickup** will be decreased by 1 and this won't be incremented again because the pickup is parked into permanent parking lot, and it won't be removed.
7) Parking is done. So, we can **sem_post(&inChargeforPickup)** to indicate that valet is available again.
8) The program will keep going until the permanent parking lot for both vehicle types is full. Program can be terminated early with ctrl+c or ctrl+z signals without any memory leaks.

For terminations, in **ParkingLot** struct I also have another integer named **simulationEnd**, I set this variable to 1 when all 8 auto and 4 pickup spaces in permanent parking lot are full. I also set this variable to 1 when ctrl+c or ctrl+z signals are received.

After setting this integer to 1, I post **newAutomobile** and **newPickup** semaphores so **carAttendant** functions will not wait for new cars forever and will be available to check **simulationEnd** integer. In each thread function **If simulationEnd == 1**, while loops will be quit with help of **break** and threads will be completed.

After all threads are completed, a cleanup will be done, and the program will be terminated.


```
serdar@serdar-pc:~/Desktop/hw3$ ./test
Automobile arrived. Temp spaces: 7, Real spaces: 8
Automobile valet took automobile.
^C
CTRL+C or CTRL+Z signal received. Waiting for current threads in sleep()...
Pickup arrived. Temp spaces: 3, Real spaces: 4
Valet parked automobile. Temp spaces: 8, Real spaces: 7
Simulation is ended. Cleanup successfull!
Terminating the program...
serdar@serdar-pc:~/Desktop/hw3$ ./test
Pickup arrived. Temp spaces: 3, Real spaces: 4
Pickup valet took pickup.
^Z
CTRL+C or CTRL+Z signal received. Waiting for current threads in sleep()...
Pickup arrived. Temp spaces: 2, Real spaces: 4
Valet parked pickup. Temp spaces: 3, Real spaces: 3
Simulation is ended. Cleanup successfull!
Terminating the program...
serdar@serdar-pc:~/Desktop/hw3$
```



```
serdar@serdar-pc:~/Desktop/hw3$ ./test
Pickup arrived. Temp spaces: 3, Real spaces: 4
Pickup valet took pickup.
Automobile arrived. Temp spaces: 7, Real spaces: 8
Automobile valet took automobile.
Pickup arrived. Temp spaces: 2, Real spaces: 4
Valet parked pickup. Temp spaces: 3, Real spaces: 3
Pickup valet took pickup.
Automobile arrived. Temp spaces: 6, Real spaces: 8
Valet parked automobile. Temp spaces: 7, Real spaces: 7
Automobile valet took automobile.
Pickup arrived. Temp spaces: 2, Real spaces: 3
Valet parked pickup. Temp spaces: 3, Real spaces: 2
Pickup valet took pickup.
Automobile arrived. Temp spaces: 6, Real spaces: 7
Valet parked automobile. Temp spaces: 7, Real spaces: 6
Automobile valet took automobile.
Pickup arrived. Temp spaces: 2, Real spaces: 2
Automobile arrived. Temp spaces: 6, Real spaces: 6
Valet parked pickup. Temp spaces: 3, Real spaces: 1
Pickup valet took pickup.
Automobile arrived. Temp spaces: 5, Real spaces: 6
Valet parked automobile. Temp spaces: 6, Real spaces: 5
Automobile valet took automobile.
Pickup arrived. No space. Exiting.
Valet parked pickup. Temp spaces: 4, Real spaces: 0
Automobile arrived. Temp spaces: 5, Real spaces: 5
Valet parked automobile. Temp spaces: 6, Real spaces: 4
Automobile valet took automobile.
Automobile arrived. Temp spaces: 5, Real spaces: 4
Pickup arrived. No space. Exiting.
Pickup arrived. No space. Exiting.
Valet parked automobile. Temp spaces: 6, Real spaces: 3
Automobile valet took automobile.
Automobile arrived. Temp spaces: 5, Real spaces: 3
Automobile arrived. No space. Exiting.
Valet parked automobile. Temp spaces: 6, Real spaces: 2
Automobile valet took automobile.
Automobile arrived. No space. Exiting.
Automobile arrived. No space. Exiting.
Pickup arrived. No space. Exiting.
Valet parked automobile. Temp spaces: 7, Real spaces: 1
Automobile valet took automobile.
Pickup arrived. No space. Exiting.
Automobile arrived. No space. Exiting.
Valet parked automobile. Temp spaces: 8, Real spaces: 0
Pickup arrived. No space. Exiting.
Simulation is ended. Cleanup successfull!
Terminating the program...
```