Design the parking lot system with the following requirements.

- The functions that the parking lot system can do:
- Create the parking lot.
- Add floors to the parking lot.
- Add a parking lot slot to any of the floors.
- Given a vehicle, it finds the first available slot, books it, creates a ticket, parks the vehicle, and finally returns the ticket.
- Unparks a vehicle given the ticket id.
- Displays the number of free slots per floor for a specific vehicle type.
- Displays all the free slots per floor for a specific vehicle type.
- Displays all the occupied slots per floor for a specific vehicle type.
- Details about the Vehicles:
- Every vehicle will have a type, registration number, and color.
- Different Types of Vehicles:
- Car
- Bike
- Truck
- Details about the Parking Slots:
- Each type of slot can park a specific type of vehicle.
- No other vehicle should be allowed by the system.
- Finding the first available slot should be based on:
- The slot should be of the same type as the vehicle.
- The slot should be on the lowest possible floor in the parking lot.
- The slot should have the lowest possible slot number on the floor.
- Numbered serially from 1 to n for each floor where n is the number of parking slots on that floor.
- Details about the Parking Lot Floors:
- Numbered serially from 1 to n where n is the number of floors.
- Might contain one or more parking lot slots of different types.
- We will assume that the first slot on each floor will be for a truck, the next 2 for bikes, and all the other slots for cars.
- Details about the Tickets:
- The ticket id would be of the following format:
  <parking_lot_id>_<floor_no>_<slot_no>
  Example: PR1234_2_5 (denotes 5th slot of 2nd floor of parking lot PR1234)
- We can assume that there will only be 1 parking lot. The ID of that parking lot is PR1234.

Input/Output Format

The code should strictly follow the input/output format and will be tested with provided test cases.

Input Format

Multiple lines with each line containing a command.
Possible commands:

- create_parking_lot <parking_lot_id> <no_of_floors> <no_of_slots_per_floor>
- park_vehicle <vehicle_type> <reg_no> <color>

- unpark_vehicle <ticket_id>
- display <display_type> <vehicle_type>
- Possible values of display_type: free_count, free_slots, occupied_slots
- exit

Stop taking the input when you encounter the word exit.

Output Format

Print output based on the specific commands as mentioned below.

create_parking_lot

Created parking lot with <no_of_floors> floors and <no_of_slots_per_floor> slots per floor

park_vehicle

Parked vehicle. Ticket ID: <ticket_id>

Print "Parking Lot Full" if slot is not available for that vehicle type.

unpark_vehicle

Unparked vehicle with Registration Number: <reg_no> and Color: <color>

Print "Invalid Ticket" if ticket is invalid or parking slot is not occupied.

display free_count <vehicle_type>

No. of free slots for <vehicle_type> on Floor <floor_no>: <no_of_free_slots>

The above will be printed for each floor.

display free_slots <vehicle_type>

Free slots for <vehicle_type> on Floor <floor_no>:
<comma_separated_values_of_slot_nos>

The above will be printed for each floor.

display occupied_slots <vehicle_type>

Occupied slots for <vehicle_type> on Floor <floor_no>:
<comma_separated_values_of_slot_nos>

The above will be printed for each floor.

Examples

Sample Input

```
create_parking_lot PR1234 2 6
display free_count CAR
display free_count BIKE
display free_count TRUCK
display free_slots CAR
display free_slots BIKE
display free_slots TRUCK
display occupied_slots CAR
display occupied_slots BIKE
display occupied_slots TRUCK
park_vehicle CAR KA-01-DB-1234 black
park_vehicle CAR KA-02-CB-1334 red
park_vehicle CAR KA-01-DB-1133 black
park_vehicle CAR KA-05-HJ-8432 white
park_vehicle CAR WB-45-HO-9032 white
park_vehicle CAR KA-01-DF-8230 black
park_vehicle CAR KA-21-HS-2347 red
display free_count CAR
```

```
display free_count BIKE
display free_count TRUCK
unpark_vehicle PR1234_2_5
unpark_vehicle PR1234_2_5
unpark_vehicle PR1234_2_7
display free_count CAR
display free_count BIKE
display free_count TRUCK
display free_slots CAR
display free_slots BIKE
display free_slots TRUCK
display occupied_slots CAR
display occupied_slots BIKE
display occupied_slots TRUCK
park_vehicle BIKE KA-01-DB-1541 black
park_vehicle TRUCK KA-32-SJ-5389 orange
park_vehicle TRUCK KL-54-DN-4582 green
park_vehicle TRUCK KL-12-HF-4542 green
display free_count CAR
display free_count BIKE
display free_count TRUCK
display free_slots CAR
display free_slots BIKE
display free_slots TRUCK
display occupied_slots CAR
display occupied_slots BIKE
display occupied_slots TRUCK
exit
```

Expected Output

```
Created parking lot with 2 floors and 6 slots per floor
No. of free slots for CAR on Floor 1: 3
No. of free slots for CAR on Floor 2: 3
No. of free slots for BIKE on Floor 1: 2
No. of free slots for BIKE on Floor 2: 2
No. of free slots for TRUCK on Floor 1: 1
No. of free slots for TRUCK on Floor 2: 1
Free slots for CAR on Floor 1: 4,5,6
Free slots for CAR on Floor 2: 4,5,6
Free slots for BIKE on Floor 1: 2,3
Free slots for BIKE on Floor 2: 2,3
Free slots for TRUCK on Floor 1: 1
Free slots for TRUCK on Floor 2: 1
Occupied slots for CAR on Floor 1:
Occupied slots for CAR on Floor 2:
Occupied slots for BIKE on Floor 1:
Occupied slots for BIKE on Floor 2:
Occupied slots for TRUCK on Floor 1:
Occupied slots for TRUCK on Floor 2:
```

Parked vehicle. Ticket ID: PR1234_1_4
Parked vehicle. Ticket ID: PR1234_1_5
Parked vehicle. Ticket ID: PR1234_1_6
Parked vehicle. Ticket ID: PR1234_2_4
Parked vehicle. Ticket ID: PR1234_2_5
Parked vehicle. Ticket ID: PR1234_2_6
Parking Lot Full
No. of free slots for CAR on Floor 1: 0
No. of free slots for CAR on Floor 2: 0
No. of free slots for BIKE on Floor 1: 2
No. of free slots for BIKE on Floor 2: 2
No. of free slots for TRUCK on Floor 1: 1
No. of free slots for TRUCK on Floor 2: 1
Unparked vehicle with Registration Number: WB-45-HO-9032 and Color: white
Invalid Ticket
Invalid Ticket
No. of free slots for CAR on Floor 1: 0
No. of free slots for CAR on Floor 2: 1
No. of free slots for BIKE on Floor 1: 2
No. of free slots for BIKE on Floor 2: 2
No. of free slots for TRUCK on Floor 1: 1
No. of free slots for TRUCK on Floor 2: 1
Free slots for CAR on Floor 1:
Free slots for CAR on Floor 2: 5
Free slots for BIKE on Floor 1: 2,3
Free slots for BIKE on Floor 2: 2,3
Free slots for TRUCK on Floor 1: 1
Free slots for TRUCK on Floor 2: 1
Occupied slots for CAR on Floor 1: 4,5,6
Occupied slots for CAR on Floor 2: 4,6
Occupied slots for BIKE on Floor 1:
Occupied slots for BIKE on Floor 2:
Occupied slots for TRUCK on Floor 1:
Occupied slots for TRUCK on Floor 2:
Parked vehicle. Ticket ID: PR1234_1_2
Parked vehicle. Ticket ID: PR1234_1_1
Parked vehicle. Ticket ID: PR1234_2_1
Parking Lot Full
No. of free slots for CAR on Floor 1: 0
No. of free slots for CAR on Floor 2: 1
No. of free slots for BIKE on Floor 1: 1
No. of free slots for BIKE on Floor 2: 2
No. of free slots for TRUCK on Floor 1: 0
No. of free slots for TRUCK on Floor 2: 0
Free slots for CAR on Floor 1:
Free slots for CAR on Floor 2: 5
Free slots for BIKE on Floor 1: 3
Free slots for BIKE on Floor 2: 2,3

Free slots for TRUCK on Floor 1:
Free slots for TRUCK on Floor 2:
Occupied slots for CAR on Floor 1: 4,5,6
Occupied slots for CAR on Floor 2: 4,6
Occupied slots for BIKE on Floor 1: 2
Occupied slots for BIKE on Floor 2:
Occupied slots for TRUCK on Floor 1: 1
Occupied slots for TRUCK on Floor 2: 1

Expectations

- Make sure that you have a working and demonstrable code
- Make sure that the code is functionally correct
- Code should be modular and readable
- Separation of concern should be addressed
- Please do not write everything in a single file (if not coding in C/C++)
- Code should easily accommodate new requirements and minimal changes
- There should be a main method from where the code could be easily testable
- [Optional] Write unit tests, if possible
- No need to create a GUI

Optional Requirements

Please do these only if you've time left. You can write your code such that these could be accommodated without changing your code much.

- Keep the code extensible to add additional types of vehicles and slot types.
- Keep the code extensible to allow a different strategy of finding the first available slot.
- Keep the code extensible to allow any other type of command.
- Keep the code extensible to allow multiple parking lots. You can assume that the input/output format can be changed for that.
- Keep the system thread-safe to allow concurrent requests.