“Talk is cheap. Show me the code.” — Linus Torvalds

We genuinely believe that *“The only way to evaluate programmers is through code.”*
In this round, we’ll provide you an elementary coding problem that you need to complete within 48 hours of receiving this activity. We expect the activity to take ~3-4 hours of your valuable time.

Guidelines for this activity

- You can use any object-oriented programming language that you feel you excel at to solve the problem.
- Feel free to ask for help when you get stuck or have any questions anywhere.
- **We expect you to produce a clean, well-documented code following all good programming practices and produce maintainable + production-ready code.**
- Goes without saying, the submission should cover all edge cases possible.
- This is our version of the famous FIZZBUZZ test, so correctness is of paramount importance.
- ***The submission should contain all the information required to run the system on both Macintosh and Ubuntu, failing which there can be direct rejection without even review happening since we won’t be able to run the activity. Be as descriptive as possible there with all relevant links and everything, assume that we are setting up our computers first time for your code to run and need the complete step by step guide.***

Feel free to reach out to the person who sent you this activity or any person tagged in the email through which this activity was posted for any questions/queries.

Problem Statement

We own a parking lot that can hold up to ‘n’ cars at any given point in time. Each slot is given a number starting at one increasing with increasing distance from the entry point in steps of one. We want to create an automated ticketing system that allows our customers to use our parking lot without human intervention.

When a car enters the parking lot, we want to have a ticket issued to the driver. The ticket issuing process includes:-

1. We are taking note of the number written on the vehicle registration plate and the age of the driver of the car.
2. And we are allocating an available parking slot to the car before actually handing over a ticket to the driver (we assume that our customers are kind enough to always park in the slots allocated to them).

The customer should be allocated a parking slot that is nearest to the entry. At the exit, the customer returns the ticket, marking the slot they were using as being available.

Due to government regulation, the system should provide us with the ability to find out:-

- *Vehicle Registration numbers for all cars which are parked by the driver of a certain age,*
- *Slot number in which a car with a given vehicle registration plate is parked.*
- *Slot numbers of all slots where cars of drivers of a particular age are parked.*

We interact with the system via a file-based input system, i.e. it should accept a filename as an input. The file referenced by filename will contain a set of commands separated by a newline, we need to execute the commands in order and produce output.

Sample Input file:-
https://gist.github.com/tarungarg546/6200f936f2208bad5d9d0e053d773489

Description of each command from the above input file:-

Command	Description of command
Create_parking_lot 6	Create a parking lot of 6 slots
Park KA-01-HH-1234 driver_age 21	Park car with vehicle registration number " KA-01-HH-1234", and the vehicle is driven by the driver of age 21
Park PB-01-HH-1234 driver_age 21	Park car with vehicle registration number " PB-01-HH-1234", and the car is driven by the driver of age 21
Slot_numbers_for_driver_of_age 21	Return all **Slot Number(Comma-separated)** of all cars which have drivers with age==21
Park PB-01-TG-2341 driver_age 40	Park car with vehicle registration number " PB-01-TG-2341", and the car is driven by the driver of age 40
Slot_number_for_car_with_number PB-01-HH-1234	Return slot number for the car with registration number "PB-01-HH-1234"
Leave 2	Vacate the slot number 2 from the parking lot, i.e. car which was parked at slot number 2 has left the space if there exists no car at slot number 2, print "Slot already vacant"
Park HR-29-TG-3098 driver_age 39	Park car with vehicle registration number " HR-29-TG-3098", and the car is driven by the driver of age 39
Vehicle_registration_number_for_driver_of_age 18	Get all parked vehicle registration number of cars parked by the driver of age 18

Output(In stdout) :-
https://gist.github.com/tarungarg546/697334d7990f758ac77a8accdf3efd98

Created parking of 6 slots
Car with vehicle registration number “KA-01-HH-1234” has been parked at slot number 1
Car with vehicle registration number “PB-01-HH-1234” has been parked at slot number 2
1,2

Car with vehicle registration number “PB-01-TG-2341” has been parked at slot number 3

2

Slot number 2 vacated, the car with vehicle registration number “PB-01-HH-1234” left the space, the driver of the car was of age 21

Car with vehicle registration number “HR-29-TG-3098” has been parked at slot number 2

null

How will the evaluation work?

---

1. ***Problem Solving*** - How effective are you at understanding the problem & designing a solution for it?
2. ***Design*** - How do you design/layout your code? Is it easily readable with nicely written interfaces, etc.?
3. ***Correctness*** - Do you explicitly think about the accuracy of your code. How well do you cover all the cases?
4. ***Communication*** - Do you clearly explain your thoughts? How well do you respond to feedback/suggestions? Do you seek help/guidance when stuck?

Remember, you are encouraged to clarify any doubts you might have over the phone directly. Please don't shy away from confirming assumptions that you are doubtful about.