# Coding Assignment (APB – Data Science Vertical)

Q1: Design a parking lot in Python.

Following objects needs to be considered in the design:

- **ParkingLot** - A parking lot is made up of 'n' number of levels/floors and 'm' number of slots per floor.
- **Levels** - Each level is an independent entity with a floor number, its lanes and the slots within it. The number of lanes are designed based on the number of slots. 10 slots make one lane
- **Slots** - The slots are considered as the independent entities to each other where in the type of the vehicle is considered to fill the slot.
- **Vehicles** - Object with plate no., company name and their type. A vehicle has the attributes of license plate and the company it is from.

Consider levels and slots as entities that are independent so that any level can be added with a desired number of slots later. Each time a vehicle comes in or goes out, a list of vehicles for the particular company is updated. Also, the available slots needs to be updated in the particular level.

Operations needs to be considered:

- **ParkVehicle** - This operation inserts a vehicle accordingly, also takes care of what company vehicle it is.
- **LeaveOperation** - This operation exits a vehicle 'C' in a level 'm'.
- **CompanyParked** - This operation allows the user to view the list of vehicles parked for a particular company.

Output must contain following items:

- Process brief
- Codebase
- Class diagram
- Work flow diagram

Be precise and just consider functionalities which are mentioned above.

Q2: Given a string S. The task is to print all permutations of a given string.

**Input:**
The first line of input contains an integer T, denoting the number of test cases. Each test case contains a single string S in capital letter.

**Output:**
For each test case, print all permutations of a given string S with single space and all permutations should be in lexicographically increasing order.

**Constraints:**
1 ≤ T ≤ 10
1 ≤ size of string ≤ 5

**Example:**

**Input:**
```
2
ABC
ABSG
```

**Output:**
```
ABC ACB BAC BCA CAB CBA
ABGS ABSG AGBS AGSB ASBG ASGB BAGS BASG BGAS BGSA BSAG BSGA GABS
GASB GBAS GBSA GSAB GSBA SABG SAGB SBAG SBGA SGAB SGBA
```