Rules of the Game

1. You have to solve the problem in any language you are comfortable with.. Your solution must be built+run on Linux.
2. Additionally, it's a huge plus point if you test drive your code.
3. Please ensure that you follow the syntax and formatting of both the input and output samples.
4. We are really, really interested in your implementation skills, so please solve the problem keeping this in mind.
5. Please ensure that the coding conventions, directory structure and build approach of your project follow the conventions set by popular open source projects in the language that you're using.
6. When implementing this solution, please use Git for version control. We expect you to send us a zip/tarball of your source code when you're done that includes Git metadata (the .git folder) in the tarball so we can look at your commit logs and understand how your solution evolved.
7. Please do not make either your solution or this problem statement publicly available by, for example, using github or bitbucket or by posting this problem to a blog or forum.

Problem Statement

I own a multi-storey parking lot that can hold up to 'n' cars at any given point in time. Each slot is given a number starting at 1 increasing with increasing distance from the entry point in steps of one. I want to create an automated ticketing system that allows my customers to use my parking lot without human intervention.

When a car enters my parking lot, I want to have a ticket issued to the driver. The ticket issuing process includes us documenting the registration number (number plate) and the colour of the car and allocating an available parking slot to the car before actually handing over a ticket to the driver (we assume that our customers are nice enough to always park in the slots allocated to them). The customer should be allocated a parking slot which is nearest to the entry. At the exit the customer returns the ticket which then marks the slot they were using as being available.

This application should let me:

- ➔ Create the Parking Lot based on the number of parking spaces available taken as N in an input field. e.g. 100 Parking spaces => N = 100
- ➔ Generate the initial number of car details.
  - ◆ e.g. Takes input m = no. of cars currently in the parking lot and generates the car Registration numbers and color randomly but with the following format:
  - ◆ e.g. KA-01-HH-1234 White or KA-04-TY-3469 Blue
  - ◆ Please follow the above Registration number format and use 4 colors (Black, White, Blue, Red)
  - ◆ Also alots them slots (ranging from 1-N)

- ➔ Shows the list of cars along with their details in a Table. e.g.
  Registration No. | Color | Slot No.
- ➔ Allows me to select a car from the table and remove it from parking the slot making that slot no. available for the next car to enter the parking slot.
  - ◆ e.g. This could be done using a button adjacent to the Car row in the Table. You can also have a function which inputs car+slot details to remove it.

- ➔ Allows me to enter the car details of the incoming car manually but in the format mentioned above and the system automatically alot them to the nearest empty slot. (nearest being slot 1 in slots 1-N)
    - ◆ e.g. if slots 4 and 19 are available, the system alots the car slot 4.

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

    A. Registration numbers of all cars of a particular colour.
    B. Slot number in which a car with a given registration number is parked. c) Slot numbers of all slots where a car of a particular colour is parked.

Please include error/warning alerts if and where applicable.