## Problem Statement

I 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 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.

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

\* Registration numbers of all cars of a particular colour.

\* Slot number in which a car with a given registration number is parked.

\* Slot numbers of all slots where a car of a particular colour is parked.

We interact with the system via a simple set of commands which produce a specific

output. Please take a look at the example below, which includes all the commands

you need to support - they're self explanatory. The system should allow input in two

ways. Just to clarify, the same codebase should support both modes of input - we

don't want two distinct submissions.

1. It should provide us with an interactive command prompt based shell where

commands can be typed in

1. It should accept a filename as a parameter at the command prompt and read the

commands from that file

### Example: File

#### Input (contents of file):

create\_parking\_lot 6

park KA-01-HH-1234 White

park KA-01-HH-9999 White

park KA-01-BB-0001 Black

park KA-01-HH-7777 Red

park KA-01-HH-2701 Blue

park KA-01-HH-3141 Black

leave 4

status

park KA-01-P-333 White

park DL-12-AA-9999 White

registration\_numbers\_for\_cars\_with\_colour White

slot\_numbers\_for\_cars\_with\_colour White

slot\_number\_for\_registration\_number KA-01-HH-3141

slot\_number\_for\_registration\_number MH-04-AY-1111

#### Output (to STDOUT)

Created a parking lot with 6 slots

Allocated slot number: 1

Allocated slot number: 2

Allocated slot number: 3

Allocated slot number: 4

Allocated slot number: 5

Allocated slot number: 6

Slot number 4 is free

Slot No. Registration No Colour

1 KA-01-HH-1234 White

2 KA-01-HH-9999 White

3 KA-01-BB-0001 Black

5 KA-01-HH-2701 Blue

6 KA-01-HH-3141 Black

Allocated slot number: 4

Sorry, parking lot is full

KA-01-HH-1234, KA-01-HH-9999, KA-01-P-333

1, 2, 4

6

Not found

### Example: Interactive

Assuming a parking lot with 6 slots, the following commands should be run in

sequence by typing them in at a prompt and should produce output as described

below the command. Note that `exit` terminates the process and returns control to

the shell.

$ create\_parking\_lot 6

Created a parking lot with 6 slots

$ park KA-01-HH-1234 White

Allocated slot number: 1

$ park KA-01-HH-9999 White

Allocated slot number: 2

$ park KA-01-BB-0001 Black

Allocated slot number: 3

$ park KA-01-HH-7777 Red

Allocated slot number: 4

$ park KA-01-HH-2701 Blue

Allocated slot number: 5

$ park KA-01-HH-3141 Black

Allocated slot number: 6

$ leave 4

Slot number 4 is free

$ status

Slot No. Registration No Colour

1 KA-01-HH-1234 White

2 KA-01-HH-9999 White

3 KA-01-BB-0001 Black

5 KA-01-HH-2701 Blue

6 KA-01-HH-3141 Black

$ park KA-01-P-333 White

Allocated slot number: 4

$ park DL-12-AA-9999 White

Sorry, parking lot is full

$ registration\_numbers\_for\_cars\_with\_colour White

KA-01-HH-1234, KA-01-HH-9999, KA-01-P-333

$ slot\_numbers\_for\_cars\_with\_colour White

1, 2, 4

$ slot\_number\_for\_registration\_number KA-01-HH-3141

6

$ slot\_number\_for\_registration\_number MH-04-AY-1111

Not found

$ exit