Data Structure: VehicleState

Consider the following definition for a data structure called *VehicleState:*

```
# A VehicleState is a model, year, color, and price

data vehicleState:
    | vehicle(  model ::  String,
                year  ::  Number,
               color   ::  String,
               price   ::  Number )
end
```

To make instances of this structure, I would write:

_____________ = ____

_____________ = ____

Choose one of your above instances, and note which dot-accessors you would use to access each of its fields:

____

____

____

____

Which of the following are functions that *could be* written based on the data definition for **`VehicleState`**? Check all that apply

□

```
# same-license : VehicleState, String -> Boolean
# Consumes a VehicleState and String, produces true if the
# given VehicleState's license plate is the same as the
# given String
```

□

```
# how-old : VehicleState, Number -> Number
# consumes a VehicleState and a year. Produces the age of
# the vehicle by subtracting its year from the given year.
```

□

```
# more-expensive : VehicleState, VehicleState -> Boolean
# consumes two VehicleStates and produces true if the first
# VehicleState is more expensive than the second
```

□

```
# is-under-warranty : VehicleState -> Boolean
# Consumes a VehicleState, produces true if the given
# VehicleState has a mileage of less than 100,000 miles
```

□

```
# paint-job : VehicleState -> VehicleState
# Consumes a VehicleState and produces a VehicleState which
# is the same as the given VehicleState, but painted red
```