**<<Autonomous Driving Car Simulation Program>>**

For this program we break down the requirements in below steps, each step is specific to requirements to achieve the goal of this simulation. Below given is layman summary of the various steps that we will need to take to achieve this requirement.

**Language Chosen: Python**

1. **Field in Rectangle Shape:**

This field is the requirement to operate the car simulation.

It will be defined by its width and height.

The bottom-left coordinate is (0,0), and the top-right coordinate is (width-1, height-1). Or (9,9).

1. **Car Positioning:**

One or more cars can be added to the field.

Each car has a unique name, starting position, and direction it's facing.

The starting position is specified as (x, y) coordinates.

The direction is one of N (North), S (South), E (East), or W (West).

1. **List of Commands for the Cars:**

The cars need some commands to simulate though. The commands will be L, R, F.

L: Rotate the car 90 degrees to the left.

R: Rotate the car 90 degrees to the right.

F: Move the car forward by one grid point in the direction it's facing.

If a car tries to move beyond the boundary of the field, the command is ignored.

1. **User Interaction for Car Simulation:**

Users interact with the simulation program via a command-line interface.

Upon launching the program, users are prompted to enter the width and height of the field.

Users can add cars to the field or run the simulation.

Adding a car involves specifying its name, initial position, and commands.

Running the simulation executes the commands for all cars on the field.

**Code Structure for these Requirements:**The code structure for this simulation program will consists of 4 Classes, because use of Classes allow us to abstract away complexity by defining objects that represent real-world entities such as in this program Field, Car, Simulation etc.

Each class encapsulates its own data (attributes) and behavior (methods).

1. **Field Class:**

This class represents the rectangular field where the cars operate.

**Attributes:**

width and height: Dimensions of the field.

**Methods:**

within\_boundary(x, y): Checks if a given coordinate (x, y) is within the boundaries of the field.

1. **Car Class:**

This class represents the autonomous driving cars.

**Attributes:**

name: Unique identifier for the car.

x and y: Current position coordinates.

direction: Current direction the car is facing.

**Methods:**

move(field): Moves the car forward by one grid point in its current direction, if within the field boundaries.

rotate\_left(): Rotates the car 90 degrees to the left.

rotate\_right(): Rotates the car 90 degrees to the right.

1. **Simulation Class:**

This class manages the overall simulation.

**Attributes:**

field: Instance of the Field class representing the simulation field.

cars: Dictionary to store the cars in the simulation.

**Methods:**

add\_car(name, x, y, direction): Adds a new car to the simulation with the specified attributes.

run\_simulation(): Executes the commands for all cars in the simulation.

**main() Function:**

Entry point of the program.

**Flow:**

Prints a welcome message and prompts the user to enter the dimensions of the field.

Creates an instance of the Simulation class.

Enters a loop where the user can choose to add a car to the field or run the simulation.

Handles user input accordingly.

**======================Scenario1 simulation Result================================**

With regards to the code written for **Scenario 1** below are the simulation results for the requirement of 1st scenario.

**Screenshot 1:**

A screenshot of a computer

Description automatically generated

**In Above Test simulation I created 10 \* 10 field, then selected 1 to “Add a car to field”.**

**Entered the name of the car as “ferrari”, then entered the initial position of car in x y format as 3 4 W.**

**Then selected option 2 to “Run Simulation” where it asked to “Please enter commands for car ferrari” and I entered “FLR” (Forward, Left and Right)**

**Then it showed the current list of cars as below:  
  
- Ferrari, (3,5) N**

**It also showed the options to Start over and Exit which while selecting the option 1 to Start over it started the simulation again and when selected the option 2 for Exit the program exited with Goodbye message.**

**===================================================================================================== =Scenario2 simulation Result ======================================**

**For the scenario 2 requirement for the simulation with multiple cars, I have edited the python code to meet the requirements for scenario 2 which includes adding multiple cars and running tests with scenarios such as:  
  
Adding Multiple cars.**

**Running simulation with multiple cars.**

**Collision output if car collides.**

**Screenshot 1:**

A computer screen with white text

Description automatically generated

**Screenshot 2:**

A screenshot of a computer

Description automatically generated