1.Data structure -

```
STRUCT
 // Controls
    double throttle;   // 0..1
    double brake;      // 0..1
    double steer;      // -1..+1

    // State
    double speed_kms;
    int gear;          // -1, 0, 1..5
    double heading;    // DEGREES (-180..180)
    double x, z;

    // Engine / limits
    double max_speed_mps;
    double accel_mps2;
    double brake_mps2;
    double drag;

    // Fuel
    double fuel;
   Double idel fuel burn ;
    double fuel_burn;
} END STRUCT

/—----------Constants—----------/

dt = 0.1;
steering_rate = 60 * M_PI / 180.0;
centering_rate = 100 * M_PI / 180.0;
heading_deadzone = 0.5 * M_PI / 180.0;
throttle_increase_rate = 3.0;
 throttle_decay_rate = 2.0;
max_reverse_speed = 20;


/* ====================== Function Declarations ====================== */

void initCar(Car *car);
void initNcurses(void);
int processInput(Car *car);
void updateThrottle(Car *car, int up_pressed);
void updateSpeed(Car *car);
```

```
void updateFuel(Car *car);
void updateAutomaticGear(Car *car);
void applySpeedLimits(Car *car);
void updateSteering(Car *car);
void updatePosition(Car *car);
void renderDisplay(const Car *car);


/—————————————Function definitions ———————————————/

FUNCTION initCar(car)

    Set All fields of car to 0

    car.max_speed_mps = 100
    car.accel_mps2 =12
    car.brake_mps2 =20
    car.drag =0.12

    car.fuel =500
    car.fuel_burn = 3
    car.idle_fuel_burn = 0.1

END FUNCTION


/———————————————————————/

FUNCTION initNcurses()

    Start ncurses mode
    Disable line buffering
    Disable echo
    Enable special keys (arrows)
    Make input non-blocking
    Hide cursor

    Print control instructions on screen

END FUNCTION

/———————————————————————————————————/

FUNCTION processInput(car)
```

```
key = read key press
up_pressed = 0

car.brake = 0
car.steer = 0

IF key is 'E' or 'e' AND fuel > 0 THEN
    Toggle engine state
 IF engine turned OFF THEN
     throttle = 0
     gear = 0
END IF
END IF

IF key is UP_ARROW THEN
 up_pressed = 1
END IF

IF key is DOWN_ARROW THEN
    IF speed > 0.5 THEN
     brake = 1
  ELSE IF engine ON AND fuel > 0 THEN
        gear = -1
        Increase throttle
        Limit throttle to max 1
END IF
END IF

IF key is LEFT_ARROW THEN
    steer = -1
END IF

IF key is RIGHT_ARROW
 steer = +1
END IF

IF key is 'Q' or 'q' THEN
    RETURN -1   // Exit program
END IF

Call updateThrottle(car, up_pressed)

RETURN 0
```

```
END FUNCTION

/================================================/

FUNCTION updateThrottle(car, up_pressed)

    IF up_pressed == 1 AND car.engine_on == 1 AND car.fuel > 0 THEN

      car.throttle += (throttle_increase_rate × dt)

        IF car.throttle > 1.0 THEN
           car.throttle = 1.0
        END IF

        IF car.gear ≤ 0 THEN
            car.gear =1
        END IF

       ELSE

        Car.throttle -=  (throttle_decay_rate × dt)

        IF car.throttle < 0 THEN
           car.throttle = 0
        END IF

       END IF

  END FUNCTION



/—————————————————————————————————/

FUNCTION updateSpeed(car)

    IF car.brake == 1 AND car.speed_mps > 0 THEN
       car.speed_mps -= (car.brake_mps2 × dt)

        IF car.speed_mps < 0 THEN
           car.speed_mps = 0
        END IF
```

```
   ELSE IF car.throttle > 0 AND car.engine_on == 1 AND car.fuel > 0 THEN

      IF car.gear == −1 THEN
        direction = −1
      ELSE
        direction = +1
      END IF

      car.speed_mps +=    (car.accel_mps2 × car.throttle × direction × dt)

   ELSE
      car.speed_mps -=    (car.drag × car.speed_mps × dt)
   END IF

END FUNCTION

/—----------------------------------------------/

Function Update Fuel (car)

 If  car .engine on ==1   and  car fuel > 0
    Then
 Burn rate ←-  car.idle_fuel burn  + ( car.fuel burn * car. Throttle )

Car. fuel  ←-   car.fuel - (burn rate * dt )

If Car. Fuel <= 0
Then
  car.fuel = 0
    Car. engine on = 0
Car throttle  = 0
Endif

End if

End Function


Function update Automatic Gear (car)

If car.gear > 0
Then
  If car.speed  >90   then  car.gear =5
```

```
Else if  car. Speed  > 70
Then
   car. Gear = 4

Else if car.Speed > 50
Then
  Car. Gear = 3

Else  if car. Speed >30
Then
Car. gear =2

Else if car. Speed  > 0.1
Then
    Car gear = 1
End if

End function



Function apply speed limits (car)

If  car gear > 0 and  car speed  > car max speed
Then
     Car. speed mps  ←— car. Max speed
    End if

If  car.Gear == -1  and  car . speed  <  - max reverse speed
 Then
 Car. speed mps  ←—-  - max reverse speed
End if

If  car speed  < 0.05  and car gear != -1
 Then
    Car speed = 0
   Car gear = 0
End if

End function
```

```
FUNCTION updateSteering(car)

   IF |car.speed_mps| > 0.1 THEN

      IF car.steer ≠ 0 THEN

         car.heading_rad ← car.heading_rad +
                        (car.steer × steering_rate × dt)

         IF car.heading_rad > π THEN
            car.heading_rad ← π
         END IF

         IF car.heading_rad < −π THEN
            car.heading_rad ← −π
         END IF

      ELSE

         IF |car.heading_rad| > heading_deadzone THEN

            car.heading_rad ← car.heading_rad +
                           ( (car.heading_rad > 0 ? −1 : +1)
                             × centering_rate × dt )

         ELSE
            car.heading_rad ← 0
         END IF

      END IF

   END IF

END FUNCTION


FUNCTION updatePosition(car)

   car.y ← car.y + (car.speed_mps × cos(car.heading_rad) × dt)
   car.x ← car.x + (car.speed_mps × sin(car.heading_rad) × dt)

END FUNCTION
```

```
FUNCTION renderDisplay(car)

    IF car.gear == 0 THEN
        gear_str ← "N"
    ELSE IF car.gear == −1 THEN
        gear_str ← "R"
    ELSE
        gear_str ← STRING(car.gear)
    END IF

    Display "Engine : " + (car.engine_on ? "ON" : "OFF")
    Display "Throttle : " + (car.throttle × 100) + "%"
    Display "Speed : " + car.speed_mps
    Display "Gear : " + gear_str
    Display "Heading : " + (car.heading_rad × 180/π) + "°"
    Display "Position : X=" + car.x + " Y=" + car.y
    Display "Fuel : " + car.fuel

END FUNCTION

FUNCTION main

    DECLARE car AS Car

    CALL initCar(car)
    CALL initNcurses()

    WHILE TRUE
        result ← CALL processInput(car)

        IF result = −1 THEN
            BREAK
        END IF

        CALL updateSpeed(car)
        CALL updateFuel(car)
        CALL updateAutomaticGear(car)
        CALL applySpeedLimits(car)
        CALL updateSteering(car)
        CALL updatePosition(car)


        CALL renderDisplay(car)
```

```
        REFRESH screen
            SLEEP 0.06 seconds

    END WHILE
    CALL endwin()

    PRINT "Simulation ended"

END FUNCTION
```