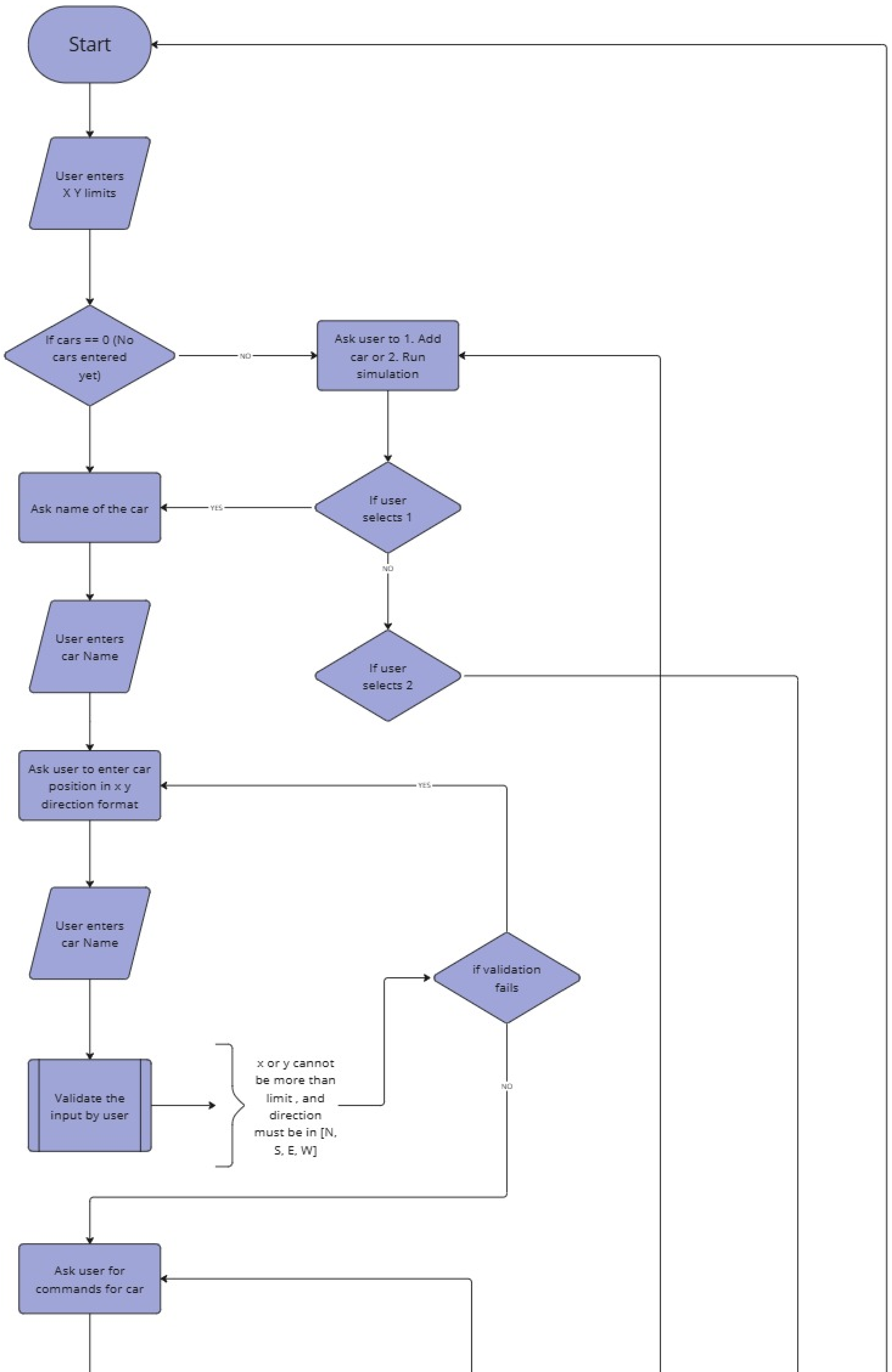
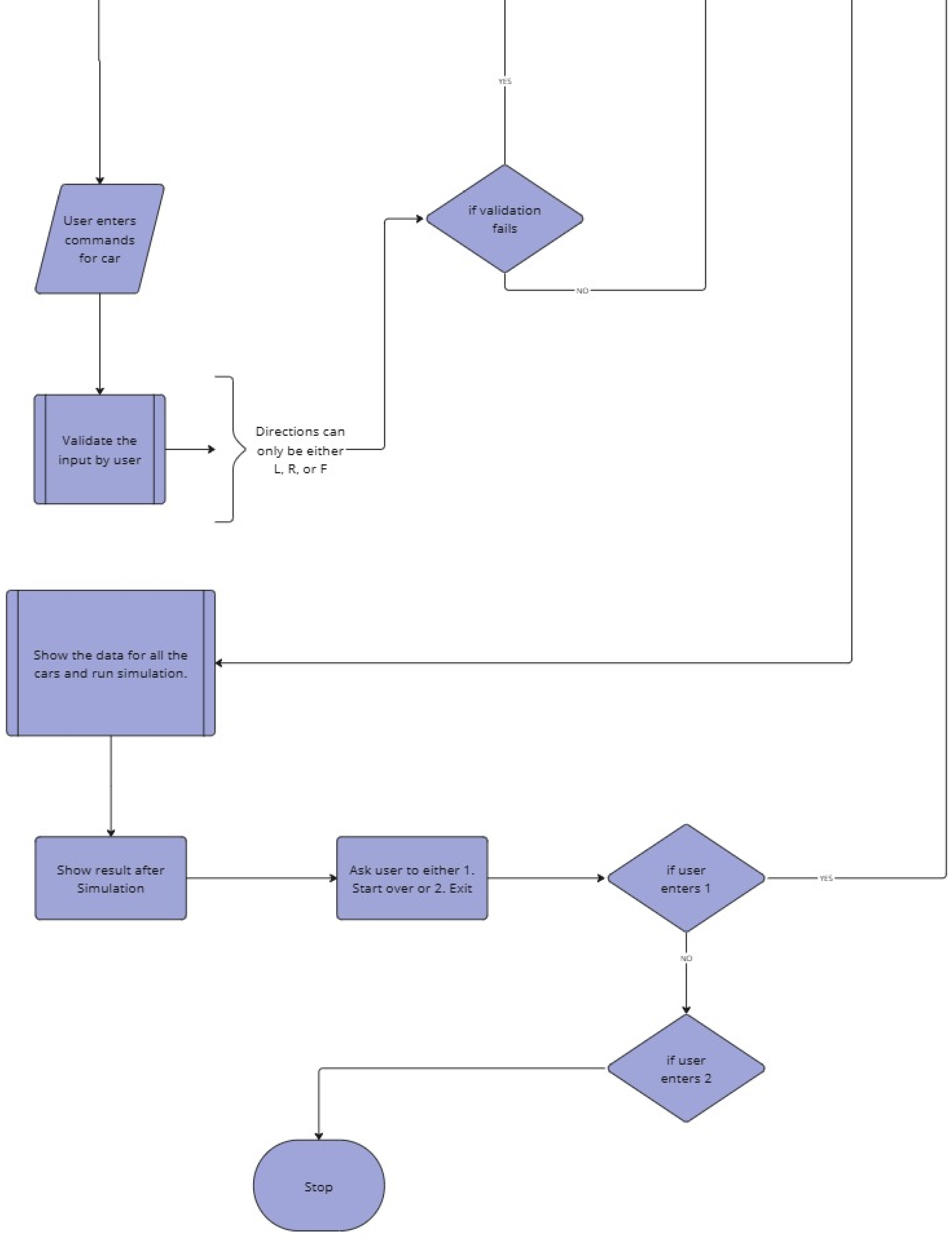

Start
User enters X Y limits
If cars == 0 (No cars entered yet)
NO
Ask user to 1. Add car or 2. Run simulation
Ask name of the car
YES
If user selects 1
NO
User enters car Name
If user selects 2
Ask user to enter car position in x y direction format
YES
User enters car Name
if validation fails
Validate the input by user
x or y cannot be more than limit , and direction must be in [N, S, E, W]
NO
Ask user for commands for car


YES
User enters commands for car
if validation fails
NO
Validate the input by user
Directions can only be either L, R, or F
Show the data for all the cars and run simulation.
Show result after Simulation
Ask user to either 1. Start over or 2. Exit
if user enters 1
YES
NO
if user enters 2
Stop

Order of the car must be preserved.

- **Make a list of car names: [A, B, C] based on the order of creation**

Order of the directions must be preserved

- **implement queue to store this , follow FIFO structure**

Create a mapping of the moves and the operations:
L = -90 degrees
R = +90 degrees

****Calculating the position at every move**

1. **Global variables will be**
   a. **(0,0) (max_x), max_y)**
   b. **list of car names according to order**
2. **For each Car entity:**
   a. **name**
   b. **pos -> x, y coordinates**
   c. **direction: current direction that the car is facing**
   d. **angle: angle of the direction.**
   e. **list of moves -> FIFO queue - can vary for each car, no limitations of length.**

**** Mapping each possible direction to X/Y axis and angle**
N = along Y axis , Forward = + 1, 0 degrees
E = along X axis , Forward = + 1, 90 degrees
W = along X axis, Forward = -1 , 270 degrees
S = along Y axis, Forward = -1, 180 degrees

```
While len(car_names_list) > 0:
  • For each car:
      • If no moves in queue,
          • remove car name from car_names_list
      • curr_pos =( (x,y) , direction, angle)
      • queue.get next move
      • if move == 'L':
      • if move == L or move == R
          • angle = (angle +/- 90) % 360
          • get direction based on angle
          • set new direction of car
      • if move == Forward
          • get curr direction
          • add or subtract 1  from X/ Y axis accordingly based on direction.
          • if above calculation results in coordinates out of range, then nullify the calculation, keep
            coordinates as-is
      • check if any of the cars collided
          • check if coordinates match with the other cars in the list of car names
          • if yes, then
              • add collision information to car entities affected
              • remove the collided cars from the list of car names
```