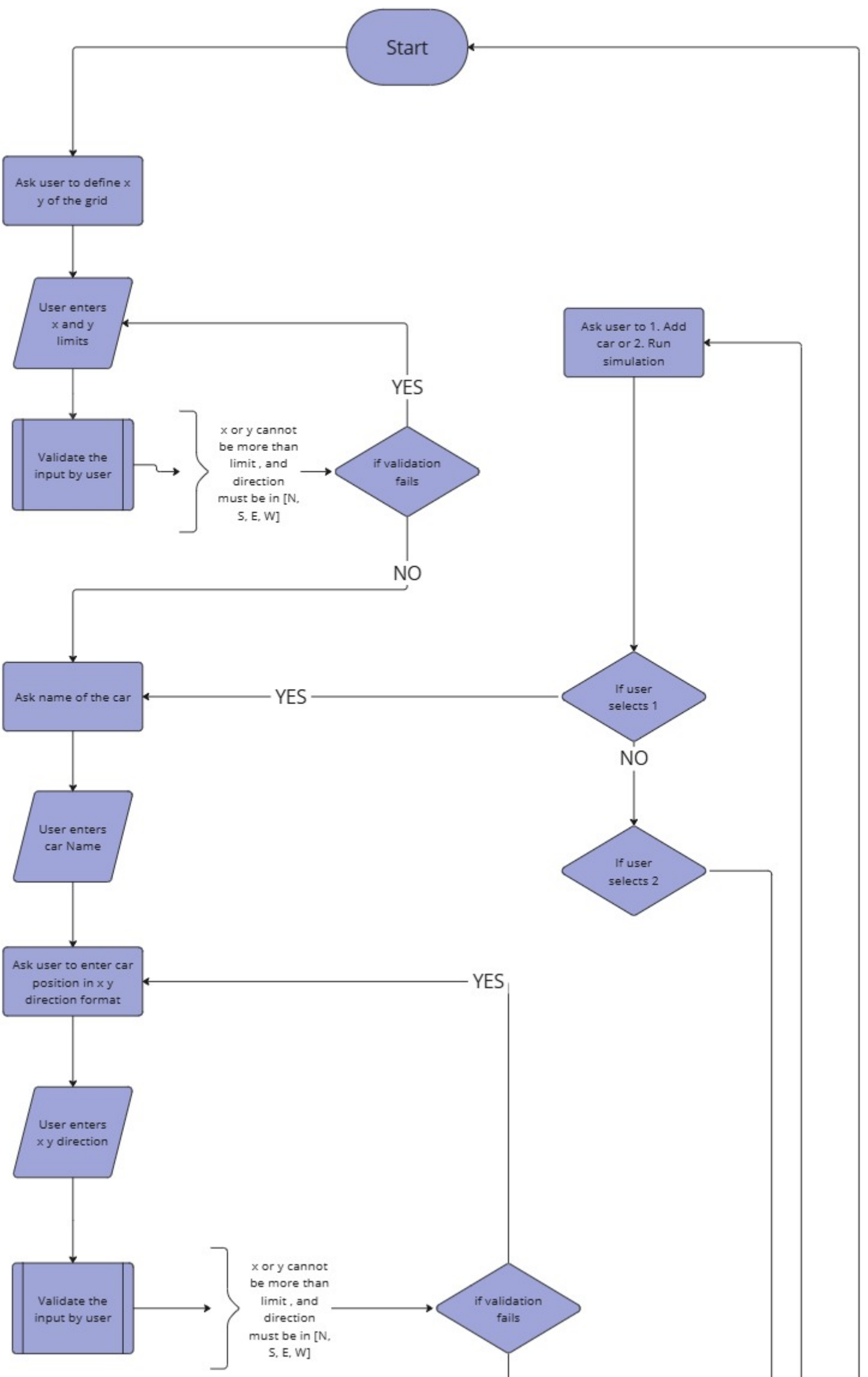
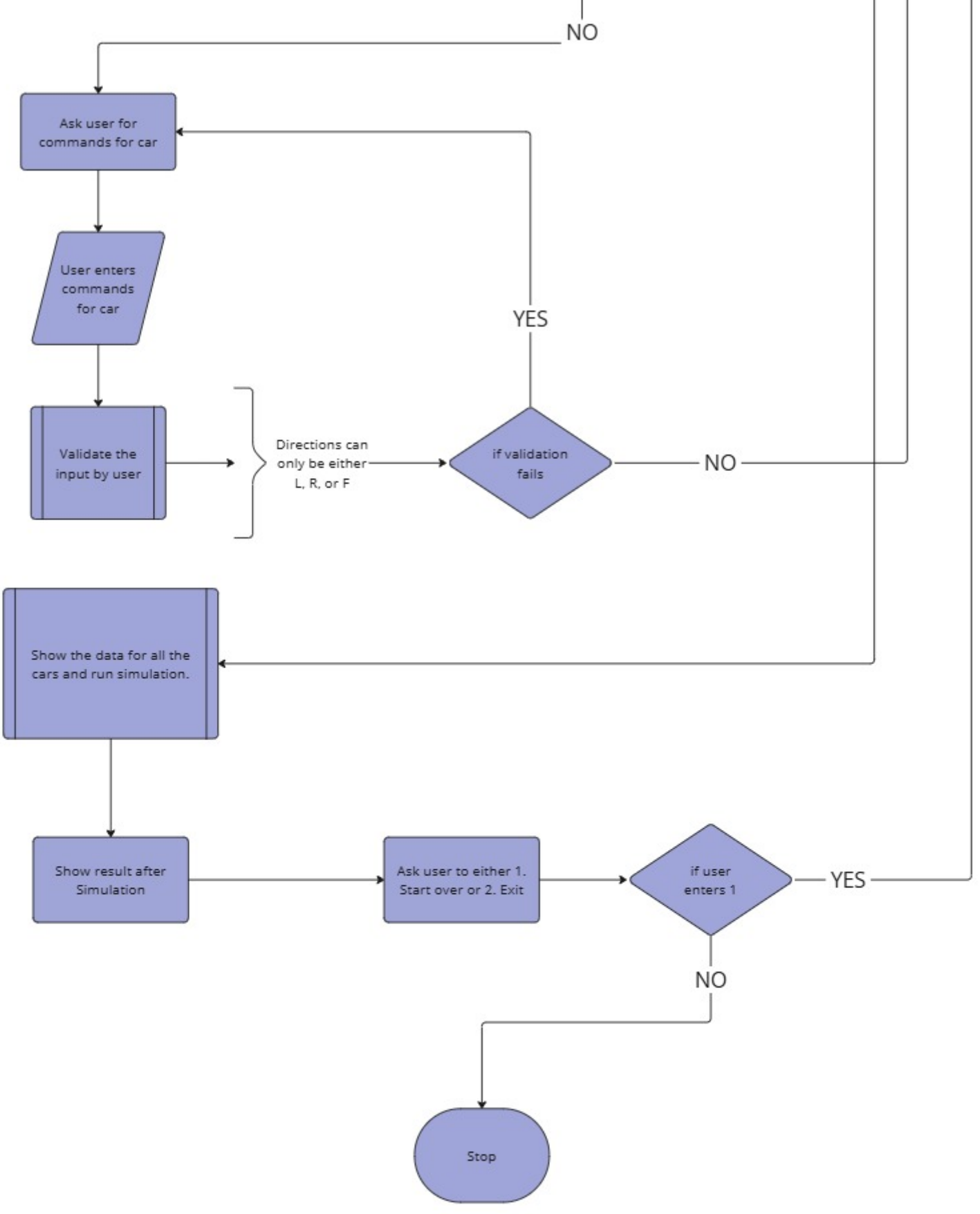

Start
Ask user to define x y of the grid
User enters x and y limits
Validate the input by user
x or y cannot be more than limit , and direction must be in [N, S, E, W]
if validation fails
YES
NO
Ask user to 1. Add car or 2. Run simulation
If user selects 1
YES
Ask name of the car
User enters car Name
Ask user to enter car position in x y direction format
YES
User enters x y direction
Validate the input by user
x or y cannot be more than limit , and direction must be in [N, S, E, W]
if validation fails
If user selects 2


NO
Ask user for commands for car
User enters commands for car
YES
Validate the input by user
Directions can only be either L, R, or F
if validation fails
NO
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
Stop

Order of the car must be preserved.

- **Make a list of cars: [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. **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:**
    - **queue.get next move**
    - **if move == L or move == R**
        - **angle = (angle +/- 90) % 360**
        - **get direction based on angle**
        - **set new direction of car**
    - **if move == F**
        - **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*

After every iteration of the cars list:

- **If no moves in queue,**
    - **remove car name from cars_list**

- **check if any of the cars collided**
    - **if car is in cars_list:**
        - check if coordinates match with the other cars in the list of car names
        - **if yes, then**
            - **set collision to true**
            - **add collision information to car entities affected**
            - **remove car from cars_list**