Problem 1

*Jurassic Park* consists of a dinosaur museum, and a park for safari riding. There are \( N \) single-passerger cars and \( M \) visitors. Each visitor wanders around the museum for a while, and then lines up to take a ride in a safari car. When a car is available, it loads the one passenger it can hold; waits for the visitor to signal he/she is ready to start the ride; and travels around the park for a random amount of time before returning to the museum. If the \( N \) cars are all being used, a visitor who wants to ride must wait; if a car is ready to load but there are no waiting visitors, then the car must wait. After the ride in the park, the car signals the visitor when it is safe to exit the car, and the visitor leaves the park.

The algorithm skeleton below simulates the above scenario. Note that the *Observer* process should be able to determine accurately the number of cars that are moving through the park at any random time. Complete the algorithm below using semaphores to synchronize the \( M \) passenger processes and the \( N \) car processes. Explain the purpose of each semaphore and shared variable.

```c
Semaphore ...
int nFullCars = 0; // number of full cars

Process Visitor (i) {
    ... walk around museum ...
    ... ride around park ...
}

Process Car (j) {
    do forever {
        ... ride around park ...
    }
}

Process Observer {
    do forever {
        ... sleep for a random amount of time ...
        printf("nFullCars = \d\n", nFullCars);
    }
}
```

Problem 2
Stallings, Problem 5.21.

Problem 3
Stallings, Problem 6.5.

Problem 4
Stallings, Problem 6.16.