Now that The city of New Orleans is using a new parking meter system, it wants to carry out a survey of city streets to determine the capacity of the streets in terms of the average number of cars that can be parked in a given street. Your purpose is to simulate parking to compute the average number of total number of cars that can parked in a given street. Parking is assumed to be parallel to the street.

For the purpose of the simulation we will assume that the length of a car is between 5.0 and 15 feet inclusive.

**Input:** The length of a city block; a double number greater than zero and less than 200.0 representing feet.

The simulation will park as many cars as possible

**Generation of cars:** we will use the Java class `Random`; the method `nextDouble()` will generate a random double number which we will use to indicate the length of the car.

**Generation of parking position:** we use `Random.nextDouble()` to generate a number that will indicate where a car’s front end should end up.

**Notice:** A length of 1 feet on each end must be allowed between consecutively parked cars.

**Parking algorithm to follow:** You must implement a recursive algorithm to fill a street with cars. Having a given block length, generate a car and a parking position, if possible park car with front end at that number; after successfully parking a car, we now have two areas to continue filling with cars, that one to the left of the car, and that one to the right of the car. Proceed doing the same up to each part.

**Two possible problems:** a. Generated car does not fit in current block; if block length is greater than minimum length needed to park a car, generate a new car. b. Generated parking position can not be used to park car, generate a new position.

**Comments on the class Random:** For the implementation you may use an instance of the class `Random`, and calls to the method `nextDouble()` of this class; this method returns a random number between 0.0 and 1.0. Thus given a city block of length L, a random number between 0.0 and L is generated using `nextDouble() * L`. To generate a car of length use `nextDouble() * (MAX_CAR_LENGTH - MIN_CAR_LENGTH) + MIN_CAR_LENGTH`.

**What to implement:**
A system that allows to run several simulations for the same city block, and produce the following statistics per each simulation,

- length of street
- max size of car parked. (computed recursively)
- min size of car parked (computed recursively)
- average of cars parked.
- max space left between two cars, or a car and a corner (computed recursively)
- min space left between two cars or a car and a corner (computed recursively)
- average spaces between cars left after all block is filled.

A list of car length and its car position in the block in increasing order of position.

**What to submit:**
a. Source code of problem  
b. Test for each class used  
c. Script of execution.

Because block length and car length may be generated at random you will have to do something in your test harness to make it use certain lengths: use a mockRandom class during testing. For the script use Random.

Example of some possible test cases: a. block length cannot fit a car. b. It can only fit one car. c. It can fit two cars.