1. The process used to mock the database is simple. First, we establish the object that will mock the database. Then, we record what the database will receive, and what it should do when it receives it. We set up the mock to receive calls to a database object, and tell it that when it receives a call for getRoomOccupant, it should return a string. It then will return the string, and give the Hotel method what it expects. We can then test to see if the string was returned properly.
2. You can use lastcall to return an exception by defining an exception object, and telling the mock to throw that exception when it receives the error.
3. If the object did not return any values, you would not need to use a stub. As is, you *could* replace it with a dynamic mock, but it would be a waste of time. If it did not return objects, a dynamic mock would be better.
4. First, we must establish a mock to be the database, and within that mock we must create an appropriate structure to hold the data, a list in this case. After the list is made and filled with data, the Hotel object is created and then we set its database to be the mock. This allows us to test the room count method.
5. The way we test to see if the car is removed from the database is quite simple. We set up a database with two cars in it, remove one, and check to see that there is only 1 car in it, and that the one that is left is the one that wasn't removed.