Donnie Waters CM2906

Lab 6 Questions

1. First we created the mock database as well as strings that we would use as return values. Then by using the LastCall.Return function, we made it so that when getRoomOccupant was called, the corresponding strings were automatically returned, thus the functions didn't need to interact with a real database, isolating them.
2. There is a method called LastCall.Throw, which throws the given exception when the designated function is called. The same procedure as using .Return except we use .Throw and pass in an exception rather than a return value.
3. Yes you still need a stub. No using a DynamicMock will not work.
4. We created a stub for a database and set the stub as target's database that it would interact with. Then we created a list of ints from 0-99 and set those as the stub's Rooms list, which is what AvailableRooms interacts with, so when target calls AvailableRooms, it returns the value from our stub database. We then assert that the AvailableRooms property and the size of our list of ints are the same, assuring that our method is returning the correct number of rooms.
5. We create the two cars and add them to the locator. Then we create serviceLocator as an instance of ServiceLocator. Then we book one car and test to make sure there is only 1 remaining car to be booked and that that car is the one that we want it to be.