1. Source code: Expedia, Test Code: ExpediaTest
2. Source Code: AssemblyInfo.cs, Booking.cs, Car.cs, Flight.cs, Hotel.cs, User.cs
3. The constructer takes 3 arguments, the flight start date, end date, and number of miles you're going, and stores them in the appropriate fields, which are all private except miles can be read (but not set) from outside this class. If you invoke the Equals method on this object and compare it with another Flight object, then it returns true if the number of miles + the flight leave/return dates are all the same, implying that 2 flights that go the same distance and leave/return on the same day are the same flight. If it's compared to a non Flight object with the Equals method, then it uses the default (base) Equals method. The base price of a flight is 200 + the number of days between the return and leave dates * 20. There are a couple basic checks to make sure your flight isn't broken in the constructor.
4. TestCode: BookingTest.cs, CarTest.cs, FlightTest.cs, HotelTest.cs, UserTest.cs
5. TestThatUserInitializes(),
   TestThatUserHasZeroFrequentFlierMilesOnInit(),
   TestThatUserCanBookEverything(),
   TestThatUserHasFrequentFlierMilesAfterBooking(),
   TestThatUserCanBookAFlight(),
   TestThatUserCanBookAHotelAndACar(),
   TestThatUserHasCorrectNumberOfFrequentFlyerMilesAfterOneFlight().
6. AreEqual(something expected, something actual),
   AreNotEqual(something expected, something actual),
   AreNotSame(object expected, object actual).
7. AreEqual – tests if the two things you pass it are equal, it's overloaded like 24 times so you can test almost anything, throws an exception if they aren't equal
   AreNotEqual – tests if the two things you pass it ARE NOT equal, also overloaded 24 times or so, throws an exception if they are equal
   AreNotSame – checks to make sure that two objects do NOT refer to the same object, throws an exception if they do
8. You can use both for objects, but "AreSame" can only be used for objects, and sees if two objects are referring to the SAME object, while "AreEqual" checks if two objects have the same values for everything.
9. That the constructor gives us a non-null object when invoked.
10. $45 per night (45 * NumberOfNightsToRent)
11. Staying for 1 night costs $45, staying for 2 nights cost $90, and staying for 10 nights costs $450
12. We already have a test for this, plus then it wouldn't be a unit test as you're testing two different parts of the program.
13. Because you can't stay in the hotel for a negative number of days
14. [ExpectedException(typeof(OutOfMemoryException))]