Service Oriented Architecture

Please read the SOA slides on myCourses. For more detailed information or additional help in answering the following questions, you may find the following links to be helpful as that is where a lot of the material came from:

https://msdn.microsoft.com/en-us/library/ee658109.aspx Chapters 5 to 18.

http://electronicdesign.com/embedded/whats-difference-between-message-centric-and-data-centric-middleware

https://msdn.microsoft.com/en-us/library/bb833022.aspx (the link is expired)

Please answer the following questions:

1) Define Service Oriented Architecture. Provide some benefits of this approach.
    a. Service Oriented Architecture is computing paradigm or let alone a style of a software design. The benefits of Service Oriented Architecture are availability, reliability, and easy to maintain. Also the greatest benefit of this approach is easy service reusability.

2) What are some concerns that need to be dealt with when using Service Oriented Architectures?
    a. One concern is heavy investments into initial using Service Oriented Architectures. Service Oriented Architectures interact with every messages that are inputted into the system and thus delay the progress and performance.

3) What is a Data-Centric Service?
    a. Data-Centric Service is a type of service. They deal with holding data storage in four different general approaches: XML files, flat files, database, and tape archive. Programmers uses Data-Centric Service typically only for disparate sources.

4) What is a Logic/Message-Centric Service?
    a. Logic/Message-Centric Service is a type of service. They deal with encapsulating business logic. When I say business logic, I mean by accessing rules, updating rules, aggregation/separation of data, use of scientific algorithms, functions, or notations, and other accounting and business rules. Programmers uses Logic/Message-Centric Service typically only for disparate clients.

5) What are the 4 most common layers of a Service Oriented Architecture and what is the purpose of each? What are some design issues for each layer?
    a. The 4 most common layers of a Service Oriented Architecture are:
        i. Presentation Layer
            1. Purpose: Changing business processes that requires easy to adjust, easy to operate for many people, and needs to have variety of user interfaces.
            2. Design Issues: Develop complex graphic designs to try and fit everyone's expectations.
        ii. Service Layer
            1. Purpose: A bridge between higher and lower layers that is characterized by a number of services it uses.
            2. Design Issues: Tends to be loosely coupled here and knowledge limitations on data service.
        iii. Business Layer
            1. Purpose: Information flow between the parties to achieve a business goal.
            2. Design Issues: Authentication and authorization (impersonation) issues exists.
        iv. Data Layer
            1. Purpose: Provides a way of showing, representing, creating, and managing the data from data storages.
            2. Design Issues: Accesses across databases that should not need to know about this information and transaction mismanagement can occur.

6) What are crosscutting concerns? List 3 different ones and at least one suggestion for how to deal with each?
    a. Crosscutting concerns is an aspect of program affecting other concerns in implementations and the design.
        i. Security
            1. Security should be dealt with by separating the data from each other.
        ii. Logging
            1. Everyone's work, access, and actions should be logged in different forms.
        iii. Database Access
            1. Database Access should be handled by minimizing users' access to the database.

7) What is Tight-Coupling? What is Loose-Coupling?
    a. Tight-coupling refers to two classes heavily relying on each other to make function's work. Tight coupling isn't good because it reduces the flexibility and reusability of a code.
    b. Loose-coupling refers to two classes not needing to depend on each other make a function work and a third class can connect to either classes to

make a function work. Unlike tight-coupling, loose coupling allows space for flexibility and reusability of a code.

8) What is Cohesion?
    a. Cohesion is a design principles that I've learned in Software Design Principles and Pattern by Kenn Martinez. The class does a task and only this task very well. Cohesion makes the class easy to maintain, reuse, and keep to it simple. You want to have more cohesion.

9) What is the difference between Authentication and Authorization?
    a. Authorization decides if you should have authorization to the content while authentication can identify who the user is while accessing the content.