Lab 3

1. **What is Spring?**
   Spring is a framework that simplify the development process of the web applications, it is flexible because you can choose the xml configuration or annotations, and it is more lightweight. Also, it is a dependency injection container.

2. **What is Spring Boot?**
   Spring Boot is built on top of the Spring Framework, it provides all the features of Spring, and it is easier to use than Spring. In Spring Boot everything is auto configured based on the jar dependencies that you added. We just need to use proper configuration for utilizing a particular functionality.

3. **What is the relation between Spring platform and Spring Boot?**
   Spring Boot is a module of Spring Platform, it is built on top of Spring. Spring Boot has all the features of the Spring Framework like dependency injection and aspect-oriented programming, with minimal configuration.

4. **What is the relation between Spring platform and Spring framework?**
   The Spring Platform and Spring Framework are related because the platform provides an ecosystem for developing applications, and the framework is the core component.

5. **What is Dependency Injection and how is it done in the Spring platform/framework?**
   Dependency Injection (DI) is a design pattern used to remove the hard-coded dependencies and make an application more loosely coupled, flexible, and easier to maintain. When you have a Dependency Injection, instead of a class creating its dependencies directly, the dependencies are injected into the class from the outside.

   Spring allows to manage dependencies and perform dependency injection in different ways:
   - **XML-based Configuration:** You can define bean configurations in XML files where you specify dependencies and their relationships.
   - **Annotation-based Configuration:** Spring supports annotations like @Autowired, @Component, @Service, @Repository, etc., which can be used to indicate dependencies and let Spring automatically wire them.
   - **Java-based Configuration:** You can use Java-based configuration with annotations like @Configuration, @Bean, etc., to define beans and their dependencies programmatically.

6. **What is Inversion of Control (IoC) and how is it related to Spring?**
   Inversion of Control is a principle where the control of the object and the lifecycle management is done by the container or framework, instead of objects creating their dependencies themselves, the dependencies are provided to them by an external entity.

In Spring the IoC is the core because Spring IoC container manages the instantiation and lifecycle of objects (beans) and injects their dependencies, thus inverting the control of object creation from the application code to the Spring container.