# Lab 4 - Short Answers

## 1. What is Spring?

Spring is a Java-based open-source framework that provides comprehensive infrastructure support for developing Java applications. It helps developers build loosely coupled, testable, and maintainable applications.

Example:

* Without Spring:  
  UserService service = new UserService(new UserRepository());  
    
  With Spring:  
  @Autowired  
  private UserService service;

## 2. What is Spring Boot?

Spring Boot is a framework built on top of Spring that simplifies the setup, configuration, and deployment of Spring applications. It removes boilerplate configuration by providing auto-configuration, embedded servers, and production-ready features.

Example:

* @SpringBootApplication  
  public class MyApp {  
   public static void main(String[] args) {  
   SpringApplication.run(MyApp.class, args);  
   }  
  }

## 3. What is the relation between Spring Platform and Spring Boot?

The Spring Platform provides the core infrastructure (IoC, AOP, transaction management, etc.). Spring Boot uses this platform but adds automation and simplification to quickly start and configure Spring applications.

In short: Spring Boot = Spring Platform + Auto Configuration + Embedded Server + Starter Dependencies

## 4. What is the relation between Spring Platform and Spring Framework?

The Spring Framework is the core part of the Spring Platform. The Spring Platform is a larger ecosystem that includes the Spring Framework and other Spring projects such as Spring Boot, Spring Cloud, Spring Data, and Spring Security.

So, Spring Framework = foundation (core features), Spring Platform = framework + related tools/libraries

## 5. What is Dependency Injection (DI) and how is it done in the Spring framework?

Dependency Injection (DI) is a design pattern where the dependencies of a class are provided (injected) by an external entity rather than the class creating them itself. Spring performs DI automatically using its IoC Container.

Example (Constructor Injection):

* @Component  
  public class UserService {  
   private final UserRepository userRepository;  
    
   @Autowired  
   public UserService(UserRepository userRepository) {  
   this.userRepository = userRepository;  
   }  
  }

## 6. What is Inversion of Control (IoC) and how is it related to Spring?

Inversion of Control (IoC) is a principle where the control of object creation and management is transferred from the developer to a container or framework. In Spring, the IoC Container (like ApplicationContext) manages object creation, wiring (dependency injection), and object lifecycle.

Example:

* ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);  
  UserService service = context.getBean(UserService.class);