2. Write short answers to the following questions.

1. **What is Spring?**

- Spring is an open-source framework for building Java-based enterprise applications. It provides comprehensive infrastructure support for developing robust and maintainable applications, including features like dependency injection, aspect-oriented programming, and transaction management.

2. **What is Spring Boot?**

- Spring Boot is an extension of the Spring framework that simplifies the process of building production-ready applications. It provides conventions and templates for setting up a Spring application quickly, along with embedded servers and default configurations. Spring Boot aims to make it easy to create stand-alone, production-grade Spring-based applications.

3. **What is the relation between Spring platform and Spring Boot?**

- Spring Boot is built on top of the Spring framework. While Spring provides the foundational components and features for building applications, Spring Boot streamlines the configuration and setup of Spring applications, making it easier and faster to get started with Spring.

4. **What is the relation between Spring platform and Spring framework?**

- The Spring platform includes the Spring framework as its core component. The Spring framework provides the fundamental building blocks for developing enterprise applications, while the Spring platform encompasses a broader ecosystem of projects, including Spring Boot, Spring Security, Spring Data, and more, that enhance and extend the capabilities of the Spring framework.

5. **What is Dependency Injection and how is it done in the Spring platform/framework?**

- Dependency Injection (DI) is a design pattern that allows the components of an application to be loosely coupled by injecting their dependencies from external sources rather than creating them within the component. In Spring, DI is achieved through the use of annotations like `@Autowired` or XML-based configuration. For example:

```java
@Service
public class MyService {
    private final MyRepository repository;

    @Autowired
    public MyService(MyRepository repository) {
        this.repository = repository;
    }

    // ...
```

```
}
```

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

- Inversion of Control (IoC) is a design principle where the control over the flow of a program's execution is shifted from the program itself to a container or framework. Spring implements IoC by managing the lifecycle and dependencies of application components. Developers provide configurations, and Spring is responsible for instantiating, configuring, and managing those components. This allows for more flexible and maintainable code. IoC is a core concept in the Spring framework.