Answer the following:

1. **Explain the use of RESTful APIs in Spring Boot.**

   Ans-

   RESTful APIs are a core feature of Spring Boot, enabling communication between clients and servers in a stateless and resource-oriented manner. They allow developers to build scalable web applications and microservices with ease.

   - **Standard HTTP Methods**
     RESTful APIs use standard HTTP methods for interaction:
     GET: To retrieve data from the server (e.g., fetching user details).
     POST: To create new resources (e.g., adding a new user).
     PUT: To update existing resources (e.g., modifying user details).
     DELETE: To remove resources (e.g., deleting a user).

   - **Annotations in Spring Boot**
     Spring Boot simplifies the creation of RESTful APIs using annotations:
     1. **@RestController:** Marks the class as a RESTful controller, meaning it can handle HTTP requests and responses.
     2. **@RequestMapping or @GetMapping, @PostMapping, @PutMapping, @DeleteMapping:** Used to map HTTP requests to specific handler methods.
     3. **@PathVariable:** Binds URI template variables to method parameters (e.g., for dynamic paths like /user/{id}).
     4. @RequestBody: Maps the HTTP request body to a method parameter, typically for processing input data in JSON or XML format.

Key Uses:

- **Resource Representation:** RESTful APIs use HTTP methods (GET, POST, PUT, DELETE) to perform CRUD operations on resources, which are often represented as JSON or XML.
- **Stateless Communication:** Each request from a client to a server contains all the information needed to understand and process the request, promoting scalability.
- **Seamless Integration:** Spring Boot provides built-in support for creating RESTful APIs using annotations and features such as @RestController, simplifying API development.
- **Customization:** With tools like Spring Security, you can secure APIs and configure them for authentication and authorization.
- **Flexible Content Negotiation:** APIs in Spring Boot can handle multiple data formats (e.g., JSON, XML) using message converters.

Example :

```
@RestController
@RequestMapping("/api/cars")
public class CarController {

  @GetMapping
  public List<Car> getAllCars() {
    // Returns all cars
  }

  @PostMapping
  public Car addCar(@RequestBody Car car) {
    // Adds a new car
  }
}
```

2. What is the use of @RequestMapping?

Ans-

The @RequestMapping annotation is used to map HTTP requests to handler methods in Spring MVC and Spring Boot applications. It is a versatile annotation that allows for flexible URL patterns and request method mappings.

Key Features:

- **URL Mapping:** Maps specific URLs or URL patterns to a method or class.
- **Method Mapping:** Can be used with HTTP methods (e.g., GET, POST) to define the type of request the method handles.
- **Path Variables:** Supports dynamic parameters in URLs using {} placeholders.
- **Headers and Parameters:** Allows filtering of requests based on headers and query parameters.

Examples:

```
@RestController

@RequestMapping("/api/users")

public class UserController {


    @RequestMapping(value = "/{id}", method = RequestMethod.GET)

    public User getUserById(@PathVariable("id") Long id) {

        // Fetch user by ID

    }

}
```