Q1]

```java
1 usage
@SpringBootApplication
public class SpringPollApplication {

    public static void main(String[] args) { SpringApplication.run(SpringPollApplication.class, args); }


    @Bean
    public String getMessage1(){
        System.out.println("hey from message1");
        return "1";
    }
}
```

The Output:

hey from message1

There is a single method that will print a specific message, "hey from message1", and there are no other possible outcomes.

Q2]

```java
1 usage
@SpringBootApplication
public class SpringPollApplication {

    public static void main(String[] args) { SpringApplication.run(SpringPollApplication.class, args); }


    @Bean
    @Qualifier("1")
    public String getMessage1(){
        System.out.println("hey from message1");
        return "1";
    }
    @Bean
    public String getMessage2(@Qualifier("1") String data ){
        System.out.println("hey from message2");
        return data ;
    }
```

The Output:

hey from message1

hey from message2

The first method will run and print "hey from message1" because the second method has a @Qualifier("1") annotation inside the parameter, so it cannot run first and there are no other possible outcomes.

Q3]

```java
@Bean
@Qualifier("1")
public String getMessage1(){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

The Outputs:

First output:

hey from message1

hey from message3

hey from message2

First, the print sentence within the first method will be printed, and then the print sentence within the third method will be printed. Therefore, the second method cannot be executed because it contains @Qualifier("3") within its parameter, so the third method will be executed first, followed by the second method.

Second possibility:

hey from message 3

hey from message 2

hey from message 1

First, the third method will be printed, because the @Qualifier("1") above the first method does not mean it has priority in printing or will be printed first. And since there is no @Qualifier within the third method, it can be printed before the first method. And if the third method is executed, we can print the sentence within the second method, because the third method has been executed, and then what is inside the first method will be printed.

Q4]

```java
@Bean
@Qualifier("1")
public String getMessage1(){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

```java
@Component
public class MainController {

    1 usage
    String data;

    public MainController(@Qualifier("1") String data){
        this.data=data;
        System.out.println("hey from Main controller");
    }

}
```

The Output:

First output:

hey from message1

hey from main controller

hey from message3

hey from message2

First, the print statement within the first method will be printed. Since the first method has been executed, we can execute the print statement within the MainController because it contains @Qualifier("1") within the parameter. Then the print statement within the third method will be printed because the second method cannot be executed before the third one because it contains @Qualifier("3") within the parameter. Finally, the print statement within the second method will be printed.

Second possibility:

hey from message3

hey from message2

hey from message1

hey from Main controller

The print statement within the third method will be printed first because the @Qualifier("1") above the first method does not mean it will be executed first or has priority. Therefore, the third method may be executed first before the first method, and it also does not contain a @Qualifier within the parameter. Since the third method has been executed, the second method will be executed, and the print statement within it will be printed. After that, the first method can be executed. After that, the Main Controller can be executed, but it cannot be executed before the first method because it contains @Qualifier("1") within the parameter, so the first method must be executed first.

Third possibility:

hey from message3

hey from message1

hey from Main Controller

hey from message2

The third method can be executed and the print statement within it can be printed because it does not contain a @Qualifier within the parameter. Therefore, the first method can also be executed after the third method, because it also does not contain a @Qualifier within the parameter. Since the first method has been executed, we can then execute the MainController method, which contains @Qualifier("1"), and we can print the print statement within it. After that, the print statement within the second method, which has a @Qualifier("3") within the parameter, can be printed because the third method has already been executed.

**@Qualifier("1") ,@Qualifier("3") above the methods does not mean it will be executed first or means sequence in order.

Forth possibility:

hey from message1

hey from message3

hey from message2

hey from Main Controller

The first method can be executed and the print statement within it can be printed because it does not contain a @Qualifier within the parameter. Therefore, the third method can also be executed after the first method, because it also does not contain a @Qualifier within the parameter. Since the third method has been executed, we can then execute the third method, which contains @Qualifier("3") within the parameter, and we can print the print statement within it. After that, the print statement within the MainController method, which has a @Qualifier("1") within the parameter, can be printed because the first method has already been executed.

Q5]

```java
@Bean
@Qualifier("1")
public String getMessage1(MainController mainController){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

```java
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

1 usage
@Component
public class MainController {

    1 usage
    String data;

    public MainController(@Qualifier("2") String data){
        this.data=data;
        System.out.println("hey from Main controller");
    }

}
```

The Output:

hey from message3

hey from message2

hey from Main Controller

hey from message1

First, the print statement in the third method will be printed because we cannot execute the print statement inside the first method, as it contains the MainController within the parameter. To execute the first method, we must first execute the MainController first. However, we cannot execute the MainController before the second method, as it contains the @Qualifier("2") within the parameter.

After the third method is executed, we can then execute the second method, which contains the @Qualifier("3")within the parameter. Then, the print statement inside the MainController method that contains the @Qualifier("2") within the parameter will be executed. Finally, the first method will be executed.