1. **What is the lambda expression of Java 8?**
   A lambda expression in Java 8 is a concise way to represent an anonymous function (a function without a name). It provides a clear and concise way to implement single-method interfaces (functional interfaces) using an expression. **The syntax is:**
   (parameters) -> expression
   Or
   (parameters) -> { statements; }

2. **Can you pass lambda expressions to a method? When?**
   Yes, we can pass lambda expressions to a method when the method parameter is a functional interface.:

3. **What is the functional interface in Java 8?**
   A functional interface in Java 8 is an interface that contains exactly one abstract method. It can have multiple default or static methods. Functional interfaces are used as the types for lambda expressions.
   The @FunctionalInterface annotation is used to indicate a functional interface.
   Example:
   ```
   @FunctionalInterface
   interface MyFunctionalInterface {
      void myMethod();
   }
   ```

4. **Why do we use lambda expressions in Java?**
   Lambda expressions are used in Java to:
   - Enable functional programming.
   - Provide a clear and concise way to represent one-method interfaces.
   - Reduce boilerplate code.
   - Improve readability and maintainability of code.
   - Facilitate parallel processing and event handling.

5. **Is it mandatory for a lambda expression to have parameters?**
   No, it is not mandatory for a lambda expression to have parameters. A lambda expression can have zero or more parameters. For example:
   ```
   // No parameters
   () -> System.out.println("No parameters");
   // One parameter
   (x) -> System.out.println(x);
   // Multiple parameters
   (x, y) -> System.out.println(x + y);
   ```