Named & default arguments

What will be printed?

```
println(listOf('a', 'b', 'c').joinToString(
        separator = "", prefix = "(", postfix = ")"))
```

?

What will be printed?

```
println(listOf('a', 'b', 'c').joinToString(
        separator = "", prefix = "(", postfix = ")"))
```

(abc)

Named arguments

```
println(listOf('a', 'b', 'c').joinToString(
        separator = "", prefix = "(", postfix = ")"))
```

```
(abc)
```

Named and default arguments

```
println(listOf('a', 'b', 'c').joinToString(
        separator = "", prefix = "(", postfix = ")"))
```

```
(abc)
```

```
println(listOf(1, 2, 3).joinToString(postfix = "."))
```

```
1, 2, 3.
```

Functions: default values

```
fun displaySeparator(character: Char = '*', size: Int = 10) {
    repeat(size) {
        print(character)
    }
}
```

```
displaySeparator('#', 5)      // #####
displaySeparator('#')         // ##########
displaySeparator()            // **********
```

Functions: named arguments

```
fun displaySeparator(character: Char = '*', size: Int = 10) {
    repeat(size) {
        print(character)
    }
}
```

```
displaySeparator(size = 5)   // *****
```

What will be printed?

```
fun displaySeparator(character: Char = '*', size: Int = 10) {
    repeat(size) {
        print(character)
    }
}

displaySeparator(3, '5')
```

1. 33333
2. 555
3. the code won’t compile

?

What will be printed?

```
fun displaySeparator(character: Char = '*', size: Int = 10) {
    repeat(size) {
        print(character)
    }
}
```

1. 33333
2. 555
3. the code won't compile

```
displaySeparator(3, '5')
```

```
displaySeparator(size = 3, character = '5')  // 555
```

Java solution: overloads

```java
public void displaySeparator(char character, int size) {
    /* ... */
}

public void displaySeparator(char character) {
    displaySeparator(character, 10);
}

public void displaySepatator() {
    displaySeparator('*');
}
```

Calling a function with default arguments from Java

```
fun sum(a: Int = 0, b: Int = 0, c: Int = 0)
```

UsingSum.java

```
// providing values for all arguments
sum(1, 2, 3);
```

@JvmOverloads annotation

```kotlin
@JvmOverloads
fun sum(a: Int = 0, b: Int = 0, c: Int = 0)
```

UsingSum.java

```java
// default values are used:
sum(1);
```

How many argument combinations are possible?

```
fun sum(a: Int = 0, b: Int = 0, c: Int = 0) = a + b + c
```

```
sum(a = 1, b = 2)
sum(c = 3)
...
```

?

How many argument combinations are possible?

```
fun sum(a: Int = 0, b: Int = 0, c: Int = 0) = a + b + c
```

$2^3 = \boxed{8}$

```
sum()
sum(a = 1)
sum(b = 2)
sum(c = 3)
```

```
sum(a = 1, b = 2)
sum(a = 1, c = 3)
sum(b = 2, c = 3)
sum(a = 1, b = 2, c = 3)
```

`@JvmOverloads` annotation

```
@JvmOverloads
fun sum(a: Int = 0, b: Int = 0, c: Int = 0)
```

Only 4 overloaded functions are generated:

```
public static final int sum(int a, int b, int c)
public static final int sum(int a, int b)
public static final int sum(int a)
public static final int sum()
```