



return from lambda



Puzzler. What will be printed?

```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap {  
        if (it == 0) return listOf()  
        listOf(it, it)  
    }  
}  
  
println(duplicateNonZero(listOf(3, 0, 5)))
```

1. `[[3, 3], [], [5, 5]]`
2. `[3, 3, 5, 5]`
3. `[]`





Puzzler. What will be printed?

```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap {  
        if (it == 0) return listOf()  
        listOf(it, it)  
    }  
}  
  
println(duplicateNonZero(listOf(3, 0, 5)))
```


1. `[[3, 3], [], [5, 5]]`

2. `[3, 3, 5, 5]`

3. `[]`

Return from function or lambda?

```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap {  
        if (it == 0) return listOf()  
        listOf(it, it)  
    }  
}
```



return from function marked with **fun**

Return empty list from the whole function

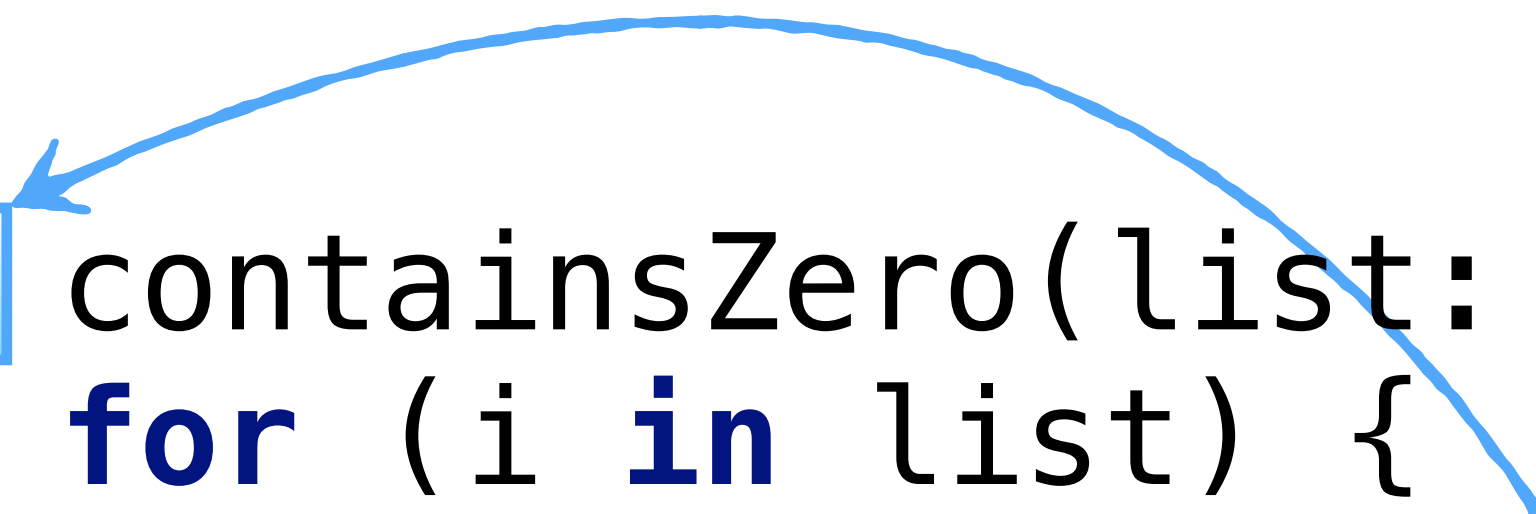
```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap {  
        if (it == 0) return listOf()  
        listOf(it, it)  
    }  
}
```

```
println(duplicateNonZero(listOf(3, 0, 5)))
```

```
[ ]
```

Why return from function?

```
fun containsZero(list: List<Int>): Boolean {  
    for (i in list) {  
        if (i == 0) return true  
    }  
    return false  
}
```



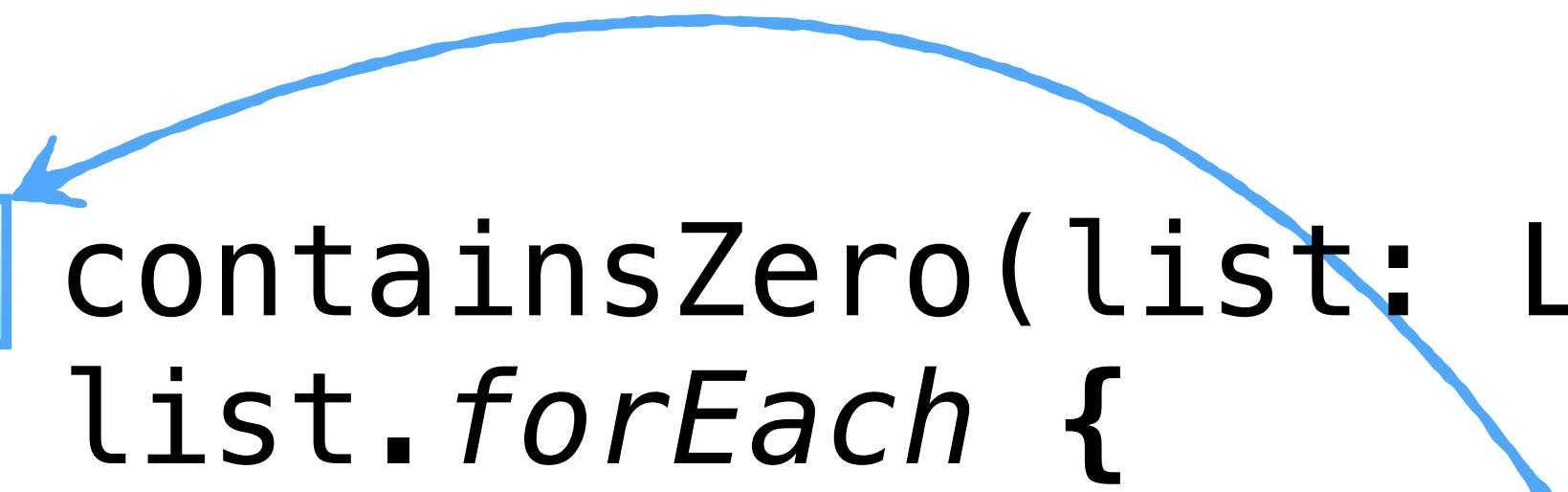
Why return from function?

```
fun containsZero(list: List<Int>): Boolean {  
    for (i in list) {  
        if (i == 0) return true  
    }  
    return false  
}
```


Why return from function?

```
fun containsZero(list: List<Int>): Boolean {  
    list.forEach {  
        if (it == 0) return true  
    }  
    return false  
}
```

Why return from function?





```
fun containsZero(list: List<Int>): Boolean {  
    list.forEach {  
        if (it == 0) return true  
    }  
    return false  
}
```

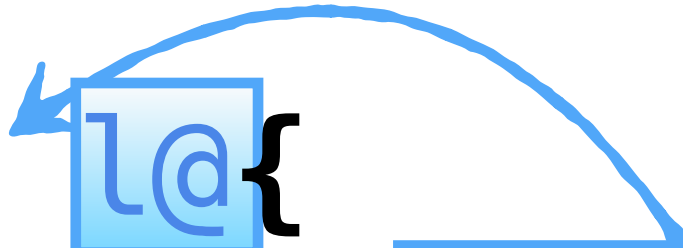
return from lambda

```
list.flatMap {  
    if (it == 0) return@flatMap listOf<Int>()  
    listOf(it, it)  
}
```

A blue curved arrow originates from the `return@flatMap` statement and points back to the opening curly brace of the lambda function. The `return@flatMap` text is highlighted with a light blue rectangular box.

return from lambda

```
list.flatMap {  
    if (it == 0)  listOf<Int>()  
    listOf(it, it)  
}
```



Solution using labels

```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap {  
        if (it == 0) return@flatMap listOf<Int>()  
        listOf(it, it)  
    }  
}
```

```
println(duplicateNonZero(listOf(3, 0, 5)))
```

[3, 3, 5, 5]

Solution using local function

```
fun duplicateNonZeroLocalFunction(list: List<Int>): List<Int> {  
    fun duplicateNonZeroElement(e: Int): List<Int> {  
        if (e == 0) return listOf()  
        return listOf(e, e)  
    }  
    return list.flatMap(::duplicateNonZeroElement)  
}
```

```
println(duplicateNonZero(listOf(3, 0, 5)))
```

[3, 3, 5, 5]

Solution using anonymous functions

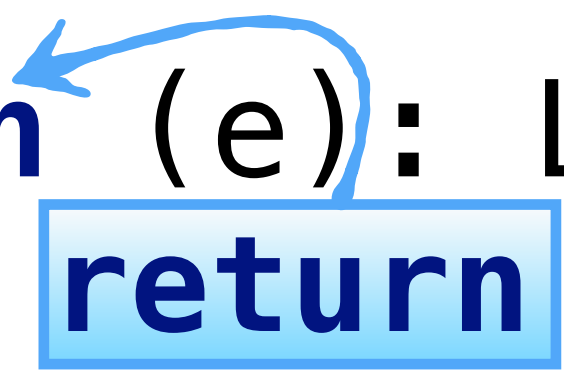
```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap(fun (e): List<Int> {  
        if (e == 0) return listOf()  
        return listOf(e, e)  
    })  
}
```

```
println(duplicateNonZero(listOf(3, 0, 5)))
```

[3, 3, 5, 5]

return from anonymous function

```
list.flatMap(fun (e): List<Int> {  
    if (e == 0) return listOf()  
    return listOf(e, e)  
})
```

A blue arrow originates from the 'return' keyword in the 'if' block and points back to the 'fun' keyword, indicating that the return statement exits the anonymous function.

Another solution: no return

```
fun duplicateNonZero(list: List<Int>): List<Int> {  
    return list.flatMap {  
        if (it == 0)  
            listOf()  
        else  
            listOf(it, it)  
    }  
}
```

```
println(duplicateNonZero(listOf(3, 0, 5)))
```

[3, 3, 5, 5]