Congratulations! You passed!

Grade received 100% | Latest Submission Grade 100% | To pass 80% or higher

Go to next item

1. Which of these are correct ways to instantiate a function type. Select all that apply. **1 / 1 point**

- [x] Using a lambda expression.

  Correct
  Correct! You can use a lambda expression to instantiate a function type.

- [x] Using instance of a user defined class that implements a function type as an interface.

  Correct
  Correct! You can instantiate a function using a defined class that implements a function type as an interface.

- [ ] Using the function name.
- [x] Using a callable reference to existing declaration using the '::' operator

  Correct
  Correct! You can use a callable reference such as '::functionName'.

2. Which of these is a correct usage of lambda expression for the given function definition? **1 / 1 point**

```
fun execute(number: Int, function: (Int) -> String) {
 println(function(number))
}
```

- ( ) `execute("Score") { "$it 100" }`
- (•) `execute(100) { "Score $it" }`
- ( ) `execute(100) ( "Score $it" )`

Correct
Correct! The above function takes in an 'Int' argument and then prints a string by concatenating it with the 'Score'.

3. Which of the these is a correct lambda expression syntax? **1 / 1 point**

- ( ) `val difference: (Int, Int) -> Int = x: Int, y: Int -> { x - y }`
- (•) `val difference: (Int, Int) -> Int = { x: Int, y: Int -> x - y }`
- ( ) `val difference: (Int, Int) -> Int = x: Int, y: Int -> x - y`

Correct
Correct! This is the correct syntax

4. Which listener interface provided by the Android framework is used to listen for a button press event? **1 / 1 point**

- (•) View.OnClickListener
- ( ) View.OnTapListener
- ( ) View.OnPressListener

Correct
Correct! The 'View' class contains an interface 'OnClickListener' that has a method 'onClick' which gets called on events such as a button press.

5. Which of these are higher-order functions? Select all that apply. **1 / 1 point**

- [ ] `fun display(x: (Int)) -> Unit`
- [x] `fun display(): (Int) -> Unit`

Correct
Correct! This is a higher-order function as it returns a function.

- [x] `fun display(x: (Int) -> Unit)`

Correct
Correct! This is a higher-order function as it takes another function as a parameter.

- [ ] `fun display(x: Int) : Unit`

6. What is the output of this code? **1 / 1 point**

```
val number = 3
var output = 2
repeat(5) { index ->
 output += (index * number)
}
println(output)

```

- ○ 47
- ○ 30
- ◉ 32

Correct
Correct! You correctly calculated the output of the given code.

7. What is the output of the following code? **1 / 1 point**

```
var sum = 0

val numberList = listOf(1, 4, 6, 7, 9)
numberList.forEach { number ->
 sum += number
}
println(sum)
```

- ○ 9
- ○ 1
- ◉ 27

Correct
Correct! The above code would iterate over each element and then add each element's value to variable named 'sum'.

8. What is the output of this code: **1 / 1 point**

```
data class Chocolate(
 val flavor: String,
 val price: Int
)
val list = listOf(
 Chocolate("Dark", 7),
 Chocolate("Milk", 4),
 Chocolate("Coffee", 2)
)
val output = list.map {
 it.flavor
}
println(output)

```

- ○ [7, 4, 2]
- ◉ [Dark, Milk, Coffee]
- ○ [Chocolate(flavor=Dark, price=7), Chocolate(flavor=Milk, price=4), Chocolate(flavor=Coffee, price=2)]

Correct
Correct! The above code transforms the initial list to a new list that contains values of 'flavor'.

9. What is the output of this code:

1 / 1 point

```
data class Chocolate(
 val flavor: String,
 val price: Int
)
val list = listOf(
 Chocolate("Dark", 7),
 Chocolate("Milk", 4),
 Chocolate("Coffee", 2)
)
val output = list.filter {
 it.price > 3
}
println(output)

```

- ○ [Chocolate(flavor=Coffee, price=2)]
- ◉ [Chocolate(flavor=Dark, price=7), Chocolate(flavor=Milk, price=4)]
- ○ [Chocolate(flavor=Dark, price=7), Chocolate(flavor=Milk, price=4), Chocolate(flavor=Coffee, price=2)]

Correct

Correct! The code above filters the 'chocolate' elements that have 'price' > 3, and returns a new list with only those elements that comply to the condition.

10. What is the output of this code:

1 / 1 point

```
val list = listOf(9, 3, 1, 6)
val output = list.fold(1) { x, y ->
 x + y
}
println(output)

```

- ○ 18
- ◉ 20
- ○ 1

Correct

Correct! The fold function accumulates a value starting from the initial value of '1' and then applies the operation to each element in the list.