Cuestionario de Scala de map, flatmap, match, filter y option en Scala.

¿Qué valor tendrá resultado?

1.-

```
val lista = List(1, 2, 3)
val resultado = lista.map(_ * 2)
```

a) List(2, 4, 6)
b) List(1, 2, 3)
c) List(2, 3, 4)
d) List(2, 6, 12

Respuesta correcta: a) List(2, 4, 6)

2.-

```
val lista = List(1, 2, 3, 4)
val resultado = lista.filter(_ % 2 == 0)
```

a) List(1, 3)
b) List(2, 4)
c) List(1, 2, 3, 4)
d) List(2)

Respuesta correcta: b) List(2, 4)

3.-

```
val opcion = Some(5)
val resultado = opcion.map(_ * 2)
```

a) Some(10)
b) Some(5)
c) None
d) 10

Respuesta correcta: a) Some(10) → se utiliza some para manejar valores opciones de manera segura. Opción es una variable de tipo Option [Int] que contiene el valor soma(5). Es decir, opción representa un valor presenta (en este caso, el número 5).

4.-

```
val lista = List(1, 2, 3)
val resultado = lista.flatMap(x => List(x, x * 2))
```

a) List(1, 2, 3, 4, 5, 6)
b) List(1, 2, 3)
c) List(1, 4, 9)
d) List(1, 2, 4, 6)

Respuesta correcta: a) List(1, 2, 3, 4, 5, 6)

5.-

```
val opcion = None
val resultado = opcion.getOrElse(10)
```

a) Some(10)
b) None
c) 10
d) Error

Respuesta correcta: c) 10 → opcion es una variable de tipo Option[Nothing] que no contiene ningún valor, es decir, None. La función getOrElse (valorAlternativo) se usa para extraer el valor de un opcion si contiene un valor por ejemplo some(x), devuelve x. Si el opcion es None devuelve el valor alternativo que en este caso es 10.

6.-

```
val opcion = Some(7)
val resultado = opcion match {
  case Some(x) => x * 2
  case None => 0
}
```

a) 14
b) 0
c) Some(14)
d) None

Respuesta correcta: a) 14 → como opcion es Some(7) se extrae el valor 7 y se multiplica por 2.

7.-

```
val lista = List(10, 20, 30)
val resultado = lista.map(x => x / 10)
```

a) List(1, 2, 3)
b) List(10, 20, 30)
c) List(0.1, 0.2, 0.3)
d) List(100, 200, 300)

Respuesta correcta: a) List(1, 2, 3)

8.-

```
val lista = List("apple", "banana", "cherry")
val resultado = lista.filter(_.startsWith("b"))
```

a) List("banana")
b) List("apple", "banana", "cherry")

c) List("b")
d) List("apple", "banana")

Respuesta correcta: a) List("banana")

9.-

```
val opcion = Some(3)
val resultado = opcion.map(x => x + 5).getOrElse(0)
```

a) 3
b) 8
c) Some(8)
d) 5

Respuesta correcta: b) 8 → opción tiene como valor presente some(3). Función map se usa para transformar el valor dentro de Some que toma el valor 3 y se suma 5, devuelve 8. Función getOrElse toma el valor 8.

10.-

```
val lista = List(1, 2, 3)
val resultado = lista.flatMap(x => if (x % 2 == 0) Some(x) else None)
```

a) List(2)
b) List(1, 2, 3)
c) List(1, 3)
d) None

Respuesta correcta: a) List(2)

11.-

```
val lista = List(1, 2, 3)

val resultado = lista.map(x => if (x % 2 == 0) Some(x) else None)
```

a) List(None, Some(2), None)
b) List(Some(2))
c) List(Some(1), Some(2), Some(3))
d) List(1, 2, 3)

Respuesta correcta: a) List(None, Some(2), None)

12.-

```
val opcion = Some(4)

val resultado = opcion.flatMap(x => Some(x * 2))
```

a) Some(8)
b) Some(4)

c) None
d) 8

Respuesta correcta: a) Some(8)

13.-

```
val lista = List(1, 2, 3, 4)

val resultado = lista.filter(_ % 2 != 0)
```

a) List(1, 3)
b) List(2, 4)
c) List(1, 2, 3, 4)
d) List(0, 1, 2, 3)

Respuesta correcta: a) List(1, 3)

14.-

```
val opcion = Some(6)

val resultado = opcion match {
 case Some(x) if x > 5 => "Mayor"
 case _ => "Menor"
}
```

a) "Mayor"
b) "Menor"
c) Some("Mayor")
d) Error

Respuesta correcta: a) "Mayor"

15.-

```
val lista = List(1, 2, 3)
val resultado = lista.map(x => x.toString)
```

a) List("1", "2", "3")
b) List(1, 2, 3)
c) List("one", "two", "three")
d) List("1", "2", "3", "4")

Respuesta correcta: a) List("1", "2", "3")

16.-

```
val lista = List(1, 2, 3)
val resultado = lista.flatMap(x => List(x * 2, x * 3))
```

a) List(1, 2, 3, 2, 4, 6)
b) List(1, 2, 3, 4, 6, 9)
c) List(2, 3, 4, 6, 6, 9)
d) List(2, 4, 6)

Respuesta correcta: b) List(2, 3, 4, 6, 6,9)

17.-

```
val lista = List("apple", "banana", "cherry")
val resultado = lista.filter(_.contains("a"))
```

a) List("apple", "banana")
b) List("apple", "banana", "cherry")
c) List("banana")
d) List("apple", "cherry")

Respuesta correcta: a) List("apple", "banana")

18.-

```
val opcion = None
val resultado = opcion.getOrElse(5)
```

a) Some(5)
b) None
c) 5
d) Error

Respuesta correcta: c) 5

19.-

```
val opcion = Some(10)
val resultado = opcion.flatMap(x => Some(x / 2))
```

a) Some(5)
b) None
c) 5
d) Some(10)

Respuesta correcta: a) Some(5)

20.-

```
val lista = List(1, 2, 3)
val resultado = lista.map(x => x + 10)
```

a) List(11, 12, 13)
b) List(1, 2, 3, 10)
c) List(1, 12, 23)
d) List(10, 20, 30)

Respuesta correcta: a) List(11, 12, 13)

21.-

```
val lista = List(10, 20, 30)
val resultado = lista.filter(_ % 10 == 0)
```

a) List(10, 20, 30)
b) List(10)
c) List(30)
d) List()

Respuesta correcta: a) List(10, 20, 30)

22.-

```
val opcion = Some(5)
val resultado = opcion.map(x => x.toString)
```

a) Some("5")
b) Some(5)
c) None
d) "5"

Respuesta correcta: a) Some("5") → opcion variable de tipo Option[Int] contiene el valor presente some (5). Función opcion.map (some(5) → some("5").

23.-

```
val lista = List(1, 2, 3, 4)
val resultado = lista.filterNot(_ % 2 == 0)
```

a) List(1, 3)
b) List(2, 4)
c) List(1, 2, 3, 4)
d) List(1)

Respuesta correcta: a) List(1, 3)

24.-

```
val opcion = Some(6)
val resultado = opcion match {
  case Some(x) if x > 5 => "Mayor"
  case _ => "Menor"
}
```

a) "Mayor"
b) "Menor"
c) Some("Mayor")
d) None

Respuesta correcta: a) "Mayor"

25.-

```
val opcion = Some(3)
val resultado = opcion.getOrElse(10)
```

a) Some(3)
b) 10
c) 3
d) None

Respuesta correcta: c) 3 → opcion variable de tipo Option[Int] que contiene Some(3).

Función getOrElse obtiene el valor que tiene opcion, contiene Some(3), devuelve el 3.

26.-

```
val lista = List(1, 2, 3)
val resultado = lista.flatMap(x => if (x % 2 == 0) Some(x) else None)
```

a) List(2)
b) List(1, 2, 3)
c) List()
d) None

Respuesta correcta: a) List(2)

27.-

```
val lista = List(5, 6, 7)
val resultado = lista.map(x => x * x)
```

a) List(25, 36, 49)
b) List(5, 6, 7)
c) List(5, 12, 21)
d) List(1, 2, 3)

Respuesta correcta: a) List(25, 36, 49)

28.-

```
val opcion = Some(4)
val resultado = opcion.getOrElse(0)
```

a) Some(4)
b) 4
c) Some(0)
d) 0

Respuesta correcta: b) 4 → misma explicación que el 25.

29.-

```
val lista = List("Scala", "Java", "Python")
val resultado = lista.filter(_.length > 4)
```

a) List("Scala", "Java")
b) List("Scala", "Python")
c) List("Java")
d) List()

Respuesta correcta: b) List("Scala", "Python")

30.-

```
val opcion = None
val resultado = opcion.getOrElse("No valor")
```

a) "No valor"
b) None
c) "Some(No valor)"
d) Error

Respuesta correcta: a) "No valor"

31.-

```
val lista = List(1, 2, 3)
val resultado = lista.flatMap(x => List(x, x + 1))
```

a) List(1, 2, 2, 3, 3, 4)
b) List(1, 3, 2, 4)
c) List(1, 2, 3)
d) List(1, 2, 3, 4, 5)

Respuesta correcta: a) List(1, 2, 2, 3, 3, 4)

32.-

```
val lista = List(10, 20, 30)
val resultado = lista.map(_.toString)
```

a) List("10", "20", "30")
b) List(10, 20, 30)
c) List(100, 200, 300)
d) List(1, 2, 3)

Respuesta correcta: a) List("10", "20", "30")

33.-

```
val opcion = Some(10)
val resultado = opcion.flatMap(x => Some(x * 2))
```

a) Some(10)
b) Some(20)
c) Some(5)
d) None

Respuesta correcta: b) Some(20)

34.-

```
val lista = List(1, 2, 3, 4)
```

```
val resultado = lista.filter(x => x % 2 == 0)
```

a) List(1, 3)
b) List(2, 4)
c) List(1, 2, 3)
d) List()

Respuesta correcta: b) List(2, 4)

35.-

```
val lista = List(1, 2, 3)
val resultado = lista.filterNot(x => x % 2 == 0)
```

a) List(2)
b) List(1, 3)
c) List(1, 2)
d) List()

Respuesta correcta: b) List(1, 3)

36.-

```
val opcion = Some(8)
val resultado = opcion.getOrElse(10)
```

a) 10
b) 8
c) Some(8)
d) Some(10)

Respuesta correcta: b) 8 → misma explicación que el ejercicio 25.

37.-

```
val lista = List(1, 2, 3, 4)
val resultado = lista.map(x => x * x)
```

a) List(1, 4, 9, 16)
b) List(1, 2, 3, 4)
c) List(2, 3, 4, 5)
d) List(2, 4, 6, 8)

Respuesta correcta: a) List(1, 4, 9, 16)

38.-

```
val lista = List(2, 4, 6, 8)
val resultado = lista.map(x => x / 2)
```

a) List(2, 4, 6, 8)
b) List(1, 2, 3, 4)
c) List(0, 1, 2, 3)
d) List(2, 3, 4, 5)

Respuesta correcta: b) List(1, 2, 3, 4)

39.-

```
val lista = List(5, 6, 7)
val resultado = lista.flatMap(x => List(x * 2, x * 3))
```

a) List(5, 6, 7, 10, 12, 14, 15, 18, 21)
b) List(10, 15, 18)
c) List(10, 12, 14, 15, 18)
d) List(10, 30, 42)

Respuesta correcta: c) List(10, 12, 14, 15, 18, 21)

40.-

```
val lista = List(1, 2, 3)
val resultado = lista.map(x => x + 10).filter(_ % 2 == 0)
```

a) List(12)
b) List(11, 13)
c) List(11)
d) List(12, 14)

Respuesta correcta: a) List(12)