Quiz: Property Based Testing

18 minutes, no stress, no embarrassment, no consequences, but **alone and quietly**

Find and fix the bug in the second property test below.

```
1  class BrokenSpec extends AnyFreeSpec with PropertyChecks with Matchers {
2    // Our usual sequence Option-List. No surprises
3    def sequence[A] (aos: List[Option[A]]) : Option[List[A]] =
4      aos.foldRight[Option[List[A]]] (Some(Nil)) {
5        (oa,z) => z flatMap (l => oa map (_::l)) }
6
7    "Returns Some if the list has no failures" in {
8      implicit def arbList[A] (implicit arb: Arbitrary[List[A]]) =
9        Arbitrary[List[Option[A]]] (arb.arbitrary map { _ map (Some (_)) })
10     forAll { (l: List[Option[Int]]) =>
11       sequence(l).isDefined shouldBe true}
12   }
13
14   "Returns None if the list has one failure" in {
15     forAll { (l :List[Option[Int]]) => sequence(l).isEmpty shouldBe true}
16   }
17 }
```

Quiz: Property Based Testing

18 minutes, no stress, no embarrassment, no consequences, but **alone and quietly**

Find and fix the bug in the second property test below.

```
1     implicit def arbList[A] (implicit arb: Arbitrary[List[A]]) =
2       Arbitrary[List[Option[A]]] (arb.arbitrary map { _ map (Some (_)) })
3     forAll { (l :List[Option[Int]]) =>
4       sequence(l) shouldNot be (None)}
5   }
6
7   "Returns None if the list has one failure" in {
8     implicit def arbFailingList[A] (implicit arb :Arbitrary[List[Option[A]]]) =
9       Arbitrary[List[Option[A]]] (arb.arbitrary filter { _ exists (_.isEmpty) })
10    forAll { (l :List[Option[Int]]) => sequence(l) shouldBe None}
11  }
12
13 }
```

Have seen the problem:1pt + Have fixed the problem:1pt = Max total:2pt