Quiz: 070 Property 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  def sequence[A] (aos: List[Option[A]]) : Option[List[A]] =
2    aos.foldRight[Option[List[A]]] (Some(Nil)) {
3      (oa,z) => z flatMap (l => oa map (_::l)) }
4
5  behavior of "sequence"
6
7  it should "succeed if the list has no failures" in check {
8    implicit def arbList[A] (implicit arb: Arbitrary[List[A]]) =
9      Arbitrary[List[Option[A]]] (arb.arbitrary map { _ map (Some (_)) })
10
11   forAll { (l :List[Option[Int]]) => sequence(l).isDefined }
12 }
13
14 it should "fail if the list has one failure" in check {
15   forAll { (l :List[Option[Int]]) => sequence(l).isEmpty }
16 }
```

Quiz: 070 Property 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  it should "succeed if the list has no failures" in check {
2    implicit def arbList[A] (implicit arb :Arbitrary[List[A]]) =
3      Arbitrary[List[Option[A]]] (arb.arbitrary map { _ map (Some (_)) })
4
5    forAll { (l :List[Option[Int]]) => sequence(l).isDefined }
6  }
7
8  it should "fail if the list has one failure" in check {
9    implicit def arbFailingList[A] (implicit arb :Arbitrary[List[Option[A]]]) =
10     Arbitrary[List[Option[A]]] (arb.arbitrary filter { _ exists (_.isEmpty) })
11
12   forAll { (l :List[Option[Int]]) => sequence(l).isEmpty }
13 }
```

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