Quiz: Algebraic Data Types

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

- Write an ADT in Scala for representing binary numbers of arbitrary size.
- A binary is either zero,
- or it is one,
- or it is a digit (one, zero) followed by a shorter binary number

Quiz: Algebraic Data Types

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

- Write an ADT in Scala for representing binary numbers of arbitrary size.
- A binary is either zero,
- or it is one,
- or it is a digit (one, zero) followed by a shorter binary number

```
1 enum Binary:
2   case One
3   case Zero
4   case O (t: Binary)
5   case I (t: Binary)
```

Quiz: Algebraic Data Types

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

- Write an ADT in Scala for representing binary numbers of arbitrary size.
- A binary is either zero,
- or it is one,
- or it is a digit (one, zero) followed by a shorter binary number

```
1 enum Binary:
2   case One
3   case Zero
4   case O (t: Binary)
5   case I (t: Binary)
```

```
1 enum Binary:
2   case Empty
3   case O(t: Binary)
4   case I(t: Binary)
```

Quiz: Algebraic Data Types

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

- Write an ADT in Scala for representing binary numbers of arbitrary size.
- A binary is either zero,
- or it is one,
- or it is a digit (one, zero) followed by a shorter binary number

```scala
enum Binary:
  case One
  case Zero
  case O (t: Binary)
  case I (t: Binary)
```

```scala
enum Binary:
  case Empty
  case O(t: Binary)
  case I(t: Binary)
```

```scala
type Digit = Boolean
enum Binary:
  case Long(msd: Digit, lsds: Binary)
  case Single(lsd: Digit)
```

Quiz: Algebraic Data Types

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

- Write an ADT in Scala for representing binary numbers of arbitrary size.
- A binary is either zero,
- or it is one,
- or it is a digit (one, zero) followed by a shorter binary number

```
1 enum Binary:
2   case One
3   case Zero
4   case O (t: Binary)
5   case I (t: Binary)
```

```
1 enum Binary:
2   case Empty
3   case O(t: Binary)
4   case I(t: Binary)
```

```
1 type Digit = Boolean
2 enum Binary:
3   case Long(msd: Digit, lsds: Binary)
4   case Single(lsd: Digit)
```

- Try to represent 101, 0, 1: `I(O(One))` for the left, `I(O(I(Empty)))` for the middle
- Success? → award 4 points.
- Minor problems? → subtract a point. Don't penalize for very minor issues.
- Zero points if the solution is polymorphic, or is not recursive.