Recitation Session Solutions, October 18 2017

Ex 1.

List[Banana]	<:	List[Fruit]
List[A]	<:	List[B]
Banana => Juice	>:	Fruit => Juice
Banana => Juice	<:	Banana => Liquid
A => C		B => D
List[Banana => Liquid]	>:	List[Fruit => Juice]
List[A => D]	>:	List[B => C]
(Fruit => Juice) => Liquid	>:	(Banana => Liquid) => Liquid
(B => C) => D	>:	(A => D) => D
Fruit => (Juice => Liquid)		Banana => (Liquid => Liquid)
B => (C => D)		A => (D => D)

Ex 2.

```
def deriv(e: Expr, v: String): Expr = e match {
  case Number(_) => Number(0)
  case Var(name) => if (name == v) Number(1) else Number(0)
  case Sum(left, right) => Sum(deriv(left, v), deriv(right, v))
  case Prod(left, right) =>
    Sum(Prod(deriv(left, v), right), Prod(left, deriv(right, v)))
}
```

Ex 3

```
def simplify(expr: Expr): Expr = expr match {
  case Number(_) =>
    expr
  case Var(_) =>
    expr
  case Sum(a, b) =>
    (simplify(a), simplify(b)) match {
      case (Number(x), Number(y)) => Number(x + y)
      case (Number(0), y) => y
      case (x, Number(0)) => x
      case (x, y) => if (x == y) Prod(Number(2), x) else Sum(x, y)
    }
  case Prod(a, b) =>
    (simplify(a), simplify(b)) match {
      case (Number(x), Number(y)) => Number(x * y)
      case (Number(0), _) => Number(0)
      case (_, Number(0)) => Number(0)
      case (Number(1), y) => y
      case (x, Number(1)) => x
      case (x, y) => Prod(x, y)
    }
}
```

Note that this is a pretty open question. The above solution is still incapable of simplifying `Sum(Number(1), Sum(Var("x"), Number(2)))` into `Sum(Number(3), Var("x"))`, which is disappointing. Can you imagine ways to improve it?

Idea for further exercise:

Implement a `simplify` function which will completely simplify all expressions given to it.

Hint: Instead of trying to add more special cases in the above function, try to convert the expression into some kind of normalized form. For instance, this normalized form could be a sum of products (of numbers and variables), represented as a list of lists of expressions. Then, you could apply simplifications on this normalized form. Finally, you would have to reconstruct a single expression out of it.