Recitation Session, October 18 2017

Please do not write on this sheet of paper
And do not use laptops during the session

Variance and Pattern Matching

This week, we will work on the idea of variance, and on pattern matching. Recall that

- Lists are covariant in their only type parameter.
- Functions are contravariant in the argument, and covariant in the result.

Ex 1. Consider the following hierarchies:

```
abstract class Fruit
class Banana extends Fruit
class Apple extends Fruit
abstract class Liquid
class Juice extends Liquid
```

Consider also the following typing relationships for `A, B, C, D`:
`A <: B` and `C <: D`.

Fill in the subtyping relation between the types below. Bear in mind that it might be that neither type is a subtype of the other.

`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. The following data types represent simple arithmetic expressions:

```
abstract class Expr
case class Number(x: Int) extends Expr
case class Var(name: String) extends Expr
case class Sum(e1: Expr, e2: Expr) extends Expr
case class Prod(e1: Expr, e2: Expr) extends Expr
```

Define a function `deriv(expr: Expr, v: String): Expr` returning the expression that is the partial derivative of `expr` with respect to the variable `v`.

```
def deriv(expr: Expr, v: String): Expr = ???
```

Here's an example run of the function:

```
> deriv(Sum(Prod(Var("x"), Var("x")), Var("y")), "x")
Sum(Sum(Prod(Var("x"), Number(1)), Prod(Number(1), Var("x"))), Number(0))
```

Ex 3. Write an expression simplifier that applies some arithmetic simplifications to an expression. For example, it would turn the above monstrous result into the following expression:

```
Prod(Var("x"), Number(2))
```