Type Inference

Exercise 1

Infer the type of the expressions:

```
x => if (x > 0) x else -x
// Int => Int

x => if (x) 1 else x
// No type.

x => if (x) (y => 0) else (y => y)
// Bool => Int => Int

x => y => x(y) && y(0)
// ((Int => Bool) => Bool) => (Int => Bool) => Bool
```

Exercise 2

Infer the generalized type of the following function definitions:

```
def S(x, y, z) = (x(z))(y(z))
// def S[A, B, C](x: A => B => C, y: A => B, z: A): C

def cm(f, g) = x => f(g(x))
// def cm[A, B, C](f: B => C, g: A => B): A => C

def cr(f) = x => (y => f(x,y))
// def cr[A, B, C](f: (A, B) => C): A => B => C

def uncr(f) = p => (f(p._1))(p._2)
// def uncr[A, B, C](f: A => B => C): (A, B) => C

def pr(x, y) = c => (c(x))(y)
// def pr[A, B, C](x: A, y: B): (A => B => C) => C
```

```
def c1(p) = p(x => (y => x))
// def c1[A, B, C](p: (A => B => A) => C): C

def c2(p) = p(x => (y => y))
// def c2[A, B, C](p: (A => B => B) => C): C

def e(x, y) = c1(pr(x,y))
// def e[A, B](x: A, y: B): A

def Sb(x, y, z) = (x(z))(z(x))
// Not type: Occurs check.
```