The Church-Turing Thesis

A function on the natural numbers is computable by a human following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing machine.

Representation

Functions Can Represent Boolean Values

If all we have to work with are functions and call expressions, is there any way to represent other primitive values?

```
def py_pred(p):
    return p(True)(False)

def f_not(p):
    return lambda a:
        lambda b: p(b)(a)

Exercise:
def f_and(p, q):
    return p(q)(f)
def f_or(p, q):
    return p(t)(q)
```

Functions Can Represent Natural Numbers

If all we have to work with are functions and call expressions, is there any way to represent other primitive values?

```
def zero(s):
    return lambda x: x

def one(s):
    return lambda x: s(x)

def two(s):
    return lambda x: s(s(x))

def successor(n):
    return lambda x: lambda y: n(s(x))(y)
def add_church(m, n):
    return lambda s:
        lambda x: m(s)(n(s)(x))

def mul_church(m, n):
    return lambda s:
        m(n(s))

def pow_church(m, n):
    return n(m)
```

Lambda Calculus Notation

Note: `lambda x: f(x)` is the same as `f`
Lambda Calculus

Variables: single letters, such as `x`

Functions: Instead of `lambda x: x`, write `λx.x`; Instead of `lambda x, y: x`, write `λxy.x`

Assignment: Write `var f = ...`

Application: Instead of `f(x)`, write `(f x)`. `f(x)(y)` and `f(x, y)` are both written `(f x y)`

Follow along! [http://chenyang.co/lambda/](http://chenyang.co/lambda/)

To type `λ`, just type `\`

| var I = λx.x  | Are `I I` and `I` the same? | Are `(K I I)` and `(K I K)` the same? |
| var K = λr.x.r | Are `(K I) I` and the same? | What's `(K K) (K K)` the same as? |
| Are `(K K I)` and `K` the same? | Can you construct a 4-argument function by just calling `K & I`?

Boolean Values

Variables: single letters, such as `x`

Functions: Instead of `lambda x: x`, write `λx.x`; Instead of `lambda x, y: x`, write `λxy.x`

Assignment: Write `var f = ...`

Application: Instead of `f(x)`, write `(f x)`. `f(x)(y)` and `f(x, y)` are both written `(f x y)`

Follow along! [http://chenyang.co/lambda/](http://chenyang.co/lambda/)

To type `λ`, just type `\`

| var T = λab.a | Define and, or, and not! | Define exclusive or: |
| var F = λab.b | xor(False, False) -> False | xor(False, True) -> True |
| Are `(K K) (K K)` the same as? | xor(True, False) -> True | xor(True, True) -> False |
| Can you construct a 4-argument function by just calling `K & I`? | | |