

COMP2022: Formal Languages and Logic

2018 Semester 2, Week 3

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pro`



THE UNIVERSITY OF  
SYDNEY

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

# Assignment Project Exam Help

WARNING

This  
on be  
Cop

<https://eduassistpro.github.io>

The material in this communication may be subject  
under the Act. Any further copying or communicat  
material by you may be subject of copyright protec

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

**Do not remove this notice.**

## OUTLINE

# Assignment Project Exam Help

- ▶ Revision: Lambda Calculus
- ▶ <https://eduassistpro.github.io/>
- ▶ numbers (a different way)
- ▶ pairs
- ▶ lists
- ▶ Add WeChat edu\_assist\_pro
- ▶ Functional Programming

## WHEN ARE $\alpha$ -REDUCTIONS REQUIRED?

# Assignment Project Exam Help

If they never change the meaning, why bother?

- <https://eduassistpro.github.io>

► ... except when it does!

# Add WeChat edu\_assist\_pr

WRONG

# Assignment Project Exam Help

$$\begin{aligned} \text{Project}_1 &= \lambda x.(\lambda y.x.y)x \\ &= \lambda x.(\lambda x.x) \end{aligned}$$

<https://eduassistpro.github.io>

$\equiv \text{FALSE}$

- Add WeChat edu\_assist\_pr  
Where is the error?

WRONG

# Assignment Project Exam Help

$\lambda x.(\lambda y.x.y)x$   
=  $\lambda x.(\lambda x.x)$  (mistake!)

<https://eduassistpro.github.io/>

$\equiv \text{FALSE}$

- ▶ Add WeChat edu\_assist\_pr
- ▶ Where is the error?
- ▶ Why is it a mistake?

WRONG

# Assignment Project Exam Help

$$\begin{aligned} \text{Project Exam He} \\ &= \lambda x.(\lambda x.x) \quad (\text{mistake!}) \end{aligned}$$

<https://eduassistpro.github.io/>

$\equiv \text{FALSE}$

# Add WeChat edu\_assist\_pr

- ▶ Where is the error?
  - ▶ Why is it a mistake?
  - ▶  $x$  was bound to the *first*  $\lambda$ , but on line 2 it is not! Free variables in  $N$  should not become bound in  $M[x := N]$

## CORRECT

# Assignment Project Exam Help

(α)

<https://eduassistpro.github.io/>

$\equiv$  TRUE

# Add WeChat edu\_assist\_pr

Rule of thumb: always perform  $\alpha$  reduction

S.

- sometimes it's necessary
  - usually makes the formula easier to read too

$\eta$ -REDUCTION (ETA)

If  $x$  is not free in  $M$ , then we can write:

# Assignment Project Exam Help

Idea: a

<https://eduassistpro.github.io>

# Add WeChat edu\_assist\_pr

$\eta$ -REDUCTION (ETA)

If  $x$  is not free in  $M$ , then we can write:

# Assignment Project Exam Help

Idea: a

- <https://eduassistpro.github.io>

# Add WeChat edu\_assist\_pr

$\eta$ -REDUCTION (ETA)

If  $x$  is not free in  $M$ , then we can write:

# Assignment Project Exam Help

Idea: a

- <https://eduassistpro.github.io>

### Uses:

- ▶ It can simplify some arguments a little
    - ▶ e.g.  $\lambda x.(\lambda y.y)x = \lambda y.y$
  - ▶ It can help to convert expressions to 'point free' form (where they do not label their variables).
    - ▶ Point-free programs can be easier to reason about, but are often difficult to read.

## OUTLINE

# Assignment Project Exam Help

- ▶ Revision - Lambda Calculus
- ▶ <https://eduassistpro.github.io/>
  - ▶ numbers (a different way)
  - ▶ pairs
  - ▶ lists
- ▶ Add WeChat edu\_assist\_pro
- ▶ Functional Programming

## NECESSARY LOGICAL NOTATION: QUANTIFIERS

# Assignment Project Exam Help

Exa

- <https://eduassistpro.github.io>
  - “ $\forall x (x + 1 = 4)$ ” is false (e.g. false on  $x = 3$ )
  - “ $\exists x \forall y (xy = 0)$ ” is true (choose  $x = 0$ )
  - “ $\exists x \forall y (xy = 1)$ ” is false (whatever we choose  $x$ , we can always find a  $y$  that doesn’t work)
  - “ $\forall x \exists y (xy = 1)$ ” is true (for any  $x$ , we can choose  $y = \frac{1}{x}$ )

## COMBINATORS

# Assignment Project Exam Help

A combinator is any expression  $M$  which contains no free variables.

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## COMBINATORS

# Assignment Project Exam Help

Exa

- ▶ <https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## COMBINATORS

# Assignment Project Exam Help

Exa

- ▶ <https://eduassistpro.github.io>
- ▶  $\lambda xy.xyy$  is a combinator (all variables bo

Add WeChat edu\_assist\_pro

## COMBINATORS

# Assignment Project Exam Help

Exa

- ▶ <https://eduassistpro.github.io>
- ▶  $\lambda xy.xyy$  is a combinator (all variables bo

## Add WeChat edu\_assist\_pro

Combinators combine values into expressions  
quantifiers or explicitly defining variables.

## COMBINATOR EXAMPLES

## Standard combinators:

- # Assignment Project Exam Help

Add WeChat edu\_assist\_pr

## COMBINATOR EXAMPLES

## Standard combinators:

- $I = \lambda x.x$  (identity)
  - $K = \lambda xy.x$  (true)

We ca

- $IM = M$
  - $KMN = M$
  - $K_* MN = N$
  - $SMNL = ML(NL)$

Add We

## COMBINATOR EXAMPLES

## Standard combinators:

- $I = \lambda x.x$  (identity)  
**►**  $K = \lambda xy.x$  (true)

▶

▶

We ca

►  $IM \equiv M$

►  ~~$KMN = M$~~

## ► ~~Add~~ $K_* MN \equiv N$

- $SMNL = ML(NL)$

Interestingly, these  $\lambda$ -free combinators are sufficient to make expressions equal to any  $\lambda$  term. We will not talk about that further today though.

## SOLVING SIMPLE EQUATIONS

# Assignment Project Exam Help

(Where  $X$ ,  $F$  are expressions in the lambda calculus)

<https://eduassistpro.github.io>

# Add WeChat edu\_assist\_pr

## SOLVING SIMPLE EQUATIONS

# Assignment Project Exam Help

(Where  $X, F$  are expressions in the lambda calculus)

"The  
 $GC$

<https://eduassistpro.github.io>

Proof:

Add WeChat edu\_assist\_pro

## SOLVING SIMPLE EQUATIONS

# Assignment Project Exam Help

(Where  $X, F$  are expressions in the lambda calculus)

"The  
GC

<https://eduassistpro.github.io>

Proof:

- Let  $G = \lambda x.xa$

Add WeChat edu\_assist\_pro

## SOLVING SIMPLE EQUATIONS

# Assignment Project Exam Help

(Where  $X, F$  are expressions in the lambda calculus)

"The  
GC

<https://eduassistpro.github.io>

Proof:

- Let  $G = \lambda x.xxx$
- Then  $GX = (\lambda x.xxx)X = XXX$

## SOLVING SIMPLE EQUATIONS

# Assignment Project Exam Help

(Where  $X, F$  are expressions in the lambda calculus)

"The  
 $GC$

<https://eduassistpro.github.io>

Proof:

- Let  $G = \lambda x. xxa$
- Then  $GX = (\lambda x. xxa)X = XXX$

That was easy... But what if we need to reason about a recursive function?

## FIXED POINT COMBINATORS

# Assignment Project Exam Help

A *fixed point combinator* is a combinator which has a fixed point.

We say

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pro`  
i.e. some input  $X$  exists which, when applied again.

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

"For all  $F$ , there exists some  $X$  such that  $FX = X$ "



<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

"For all  $F$ , there exists some  $X$  such that  $FX = X$ "



Proof  
Let

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

"For all  $F$ , there exists some  $X$  such that  $FX = X$ "



Proof  
Let

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

"For all  $F$ , there exists some  $X$  such that  $FX = X$ "



Proof  
Let

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pro`

=

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

"For all  $F$ , there exists some  $X$  such that  $FX = X$ "



Proof  
Let

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pro`

$$\begin{aligned} X &= WW \\ &= (\lambda x. F(x)) W \end{aligned}$$

$$= F(WW) \qquad \beta \qquad \text{reduction}$$

=

## FIXED POINT THEOREM (I)

Theorem:

$\forall F \exists X \quad FX = X$

"For all  $F$ , there exists some  $X$  such that  $FX = X$ "



Proof  
Let

<https://eduassistpro.github.io>

$$\begin{aligned} X &= WW \\ &= (\lambda x. F(x)) W \\ &= F(WW) && \beta \text{ reduction} \\ &= FX && (\text{def. of } X) \end{aligned}$$

## FIXED POINT THEOREM (II)

There is a fixed point combinator (the “Y Combinator”)

Assignment Project Exam Help

such that

Proo <https://eduassistpro.github.io>

$YF =$

Add WeChat edu\_assist\_pro

## FIXED POINT THEOREM (II)

There is a fixed point combinator (the “Y Combinator”)

Assignment Project Exam Help

such that

Proo <https://eduassistpro.github.io>

$$YF = (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))) Y$$

Add WeChat edu\_assist\_pro

## FIXED POINT THEOREM (II)

There is a fixed point combinator (the “Y Combinator”)

Assignment Project Exam Help

such that

Proof <https://eduassistpro.github.io>

$$YF = (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \quad Y)$$

$$= (\lambda x.F(xx))(\lambda x.F(xx))F$$

=

## FIXED POINT THEOREM (II)

There is a fixed point combinator (the “Y Combinator”)

Assignment Project Exam Help

such that

Proof <https://eduassistpro.github.io>

$$YF = (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \quad Y)$$

Add WeChat edu\_assist\_pro

$$= F((\lambda x.F(xx))(\lambda x.F(xx)))$$

=

## FIXED POINT THEOREM (II)

There is a fixed point combinator (the “Y Combinator”)

# Assignment Project Exam Help

such that

Proof <https://eduassistpro.github.io>

$$YF = (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F \quad Y)$$

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

$$= F((\lambda x.F(xx))(\lambda x.F(xx)))$$

$$= F(YF) \quad (\text{by equality above})$$

## Y COMBINATOR

So, uh... Why is this interesting?

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## Y COMBINATOR

# Assignment Project Exam Help

1.

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## Y COMBINATOR

So, uh... Why is this interesting?

# Assignment Project Exam Help

1.

<https://eduassistpro.github.io>

2. Bad news: it leads to Curry's Paradox, and th

incompleteness of lambda calculus

- ▶ not all valid expressions can be proved / co

Add WeChat `edu_assist_pro`

## RECURSION EXAMPLE

Suppose we want to compute factorials:

**Assignment Project Exam Help**

$$f(n) = \text{if } (n == 0) \text{ then } 1 \text{ else } n * f(n - 1)$$

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

Suppose we want to compute factorials:

**Assignment Project Exam Help**

$$f(n) = \text{if } (n == 0) \text{ then } 1 \text{ else } n * f(n - 1)$$

We'

- <https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

Suppose we want to compute factorials:

**Assignment Project Exam Help**

$$f(n) = \text{if } (n == 0) \text{ then } 1 \text{ else } n * f(n - 1)$$

We'

► <https://eduassistpro.github.io>

- This notation, indicating  $n$  repe  
(dangerous, but convenient)

**Add WeChat edu\_assist\_pro**

- Be aware that Church numerals have the f

$$\lambda fz.f(f(f(f(fz)))) \neq \lambda fz.fffffz$$

## RECURSION EXAMPLE

Suppose we want to compute factorials:

# Assignment Project Exam Help

$$f(n) = \text{if } (n == 0) \text{ then } 1 \text{ else } n \cdot f(n - 1)$$

We'll

- ▶ <https://eduassistpro.github.io>
- ▶ Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

Suppose we want to compute factorials:

# Assignment Project Exam Help

$$f(n) = \text{if } (n == 0) \text{ then } 1 \text{ else } n \cdot f(n - 1)$$

We'll

- ▶ <https://eduassistpro.github.io>
- ▶ ISZERO :=  $\lambda n. n(\lambda x. \text{FALSE}) T$ 
  - ▶ Returns TRUE if the argument is a Church numeral  
any other Church numeral

ISZERO ZERO

# Assignment Project Exam Help

*ISZERO ZERO*

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

## ISZERO ZERO

# Assignment Project Exam Help

*ISZERO ZERO*

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

ISZERO ZERO

# Assignment Project Exam Help

## *ISZERO ZERO*

<https://eduassistpro.github.io>

—

Add WeChat edu\_assist\_pr

## ISZERO ZERO

# Assignment Project Exam Help

## *ISZERO* *ZERO*

<https://eduassistpro.github.io>

# Add WeChat edu\_assist\_pr

## ISZERO ZERO

# Assignment Project Exam Help

*ISZERO ZERO*

<https://eduassistpro.github.io>

$$= (\lambda f z. z) (\lambda x. \text{FALSE}) \text{ TRUE} \quad \text{def.ZERO}$$

$$= (\lambda z. z) \text{ TRUE} \quad (\beta)$$

Add WeChat edu\_assist\_pro

## ISZERO ZERO

# Assignment Project Exam Help

*ISZERO ZERO*

<https://eduassistpro.github.io>

$$= (\lambda f z. z) (\lambda x. \text{FALSE}) \text{ TRUE} \quad \text{def. ZERO}$$

$$\begin{aligned} &= (\lambda z. z) \text{ TRUE} \\ &= \text{TRUE} \end{aligned} \quad \begin{array}{l} (\beta) \\ (\beta) \end{array}$$

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

<https://eduassistpro.github.io>

=  
Add WeChat edu\_assist\_pro  
=

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

ISZERO ONE

ERO)

<https://eduassistpro.github.io>

# Add WeChat edu\_assist\_pr

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

ERO)

<https://eduassistpro.github.io>

=  
Add WeChat edu\_assist\_pro  
=

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

ISZERO ONE

ERO)

<https://eduassistpro.github.io/ONE/>

Add WeChat edu\_assist\_pr

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

ERO)

<https://eduassistpro.github.io/f.ONE>

$$= (\lambda z.(\lambda x.FALSE)z) \text{ TRUE}$$

(β)

Add WeChat edu\_assist\_pro

=

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

ERO)

<https://eduassistpro.github.io/f.ONE>

$$= (\lambda z.(\lambda x.FALSE)z) \text{ TRUE} \quad (\beta)$$

Add WeChat edu\_assist\_pro

=

ISZERO ONE

# Assignment Project Exam Help

*ISZERO ONE*

ERO)

<https://eduassistpro.github.io/f.ONE>

$$= (\lambda z.(\lambda x.FALSE)z) \text{ TRUE} \quad (\beta)$$

$$= (\lambda c.FALSE) \text{ TRUE} \quad (\beta)$$

$$= FALSE \quad (\beta)$$

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

Suppose we want to compute factorials:

**Assignment Project Exam Help**

We'll

- ▶ <https://eduassistpro.github.io>
- ▶ MULT :=  $\lambda xyz.x(yz)$  (seen previo

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

Suppose we want to compute factorials:

**Assignment Project Exam Help**

We'll

- ▶ <https://eduassistpro.github.io>
- ▶ MULT :=  $\lambda xyz.x(yz)$  (seen previo
- ▶ PRED :=  $\lambda n f g x (\lambda y. h(f)(y))(\lambda y.$ 
  - ▶ This gives the predecessor of a number
  - ▶ PRED 1 = 0, PRED 2 = 1, ..., PRED n = (n-1)
  - ▶ The derivation of this is *much* longer than for the operations which increase numbers

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

## RECURSION EXAMPLE

Suppose we want to compute factorials:

**Assignment Project Exam Help**

We'll

- ▶ <https://eduassistpro.github.io>
- ▶ MULT :=  $\lambda xyz.x(yz)$  (seen previo
- ▶ PRED :=  $\lambda n f g x (\lambda y u . h(f)(yf))(xy)$ .
  - ▶ This gives the predecessor of a number
  - ▶ PRED 1 = 0, PRED 2 = 1, ..., PRED n = (n-1)
  - ▶ The derivation of this is *much* longer than for the operations which increase numbers
    - ▶ Subtraction and division are also difficult!

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

PRED TWO... IS A MONSTER

$\text{PRED TWO} = \lambda nfx. n(\lambda gh.h(gf))(\lambda y.x)(\lambda u.u) \text{ TWO}$   
 $= \lambda fx. \text{ TWO } (\lambda gh.h(gf))(\lambda y.x)(\lambda u.u)$   
     $)$

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

$$\begin{aligned} &= \lambda f. (\lambda gh.h(gf))((\lambda \\ &= \lambda fx. (\lambda gh.h(gf))(\lambda h. \\ &= \lambda fx. (\lambda gh.h(gf))(\lambda i.i x)(\lambda u.u) \\ &= \lambda fx. (\lambda h.h((\lambda i.i x)f))(\lambda u.u) \\ &= \lambda fx. (\lambda h.h(fx))(\lambda u.u) \\ &= \lambda fx. (\lambda u.u)(fx)) = \lambda fx. fx = \text{ONE} \end{aligned}$$

## RECUSION EXAMPLE

# Assignment Project Exam Help

Suppose we want to compute factorials:

We'll <https://eduassistpro.github.io>

- ▶  $c_n = \lambda f x. f^n(x)$
- ▶ ISZERO :=  $\lambda n. n(\lambda x. FALSE) T$
- ▶ MULT :=  $\lambda xyz. x(yz)$
- ▶ PRED :=  $\lambda nfx. n(\lambda gh. h(gf))(\lambda y.x)(\lambda u.u)$

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

We want to write something like:

FACT( $n$ ) = ISZERO( $n$ ) . (MULT  $n$  (FACT(PRED( $n$ ))))

We can  
Com

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

We want to write something like:

FACT( $n$ ) = ( $\lambda f n.$  ISZERO( $n$ ) 1 MULT  $n$  ( $f$  (PRED( $n$ ))))"

We can  
Com

<https://eduassistpro.github.io>

- $H = \lambda fn.$  (ISZERO  $n$ ) 1 MULT  $n$  ( $f$  (PRED  $n$ ))
  - $H$  takes a function and a number. If the number is 1, it returns 1, otherwise it returns the product of the number and the result of applying  $f$  to the predecessor of the number.

Add WeChat edu\_assist\_pro

## RECURSION EXAMPLE

We want to write something like:

" $\text{FACT}(n) = (\text{ISZERO } n) \cdot 1 + \text{MULT } n \cdot (\text{FACT}(\text{PRED } n))$ "

We ca

Com

<https://eduassistpro.github.io>

- ▶  $H = \lambda fn.(\text{ISZERO } n) \cdot 1 + \text{MULT } n \cdot (f \ (\text{PRED } n))$ 
  - ▶  $H$  takes a function and a number. If the number is 0, it returns 1, otherwise it returns the product of the number and the result of applying  $f$  to its predecessor.
- ▶  $\text{FACTORIAL} = Y \ H$ 
  - ▶ Because  $YH = H(YH)$ , the Y Combinator helps us to apply the  $H$  function to itself

Add WeChat edu\_assist\_pro

## FACTORIAL 5(OVERVIEW)

- ▶  $H = \lambda fn.(ISZERO\ n\ 1\ (\text{MULT}\ n\ (f\ (\text{PRED}\ n))))$

▶  $\text{FACTORIAL} = Y\ H$

$H$  takes a function  $f$  and a number  $n$ . It returns 1 if the number is 0, oth

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FACTORIAL 5(OVERVIEW)

- ▶  $H = \lambda fn.(ISZERO\ n\ 1\ (\text{MULT}\ n\ (f\ (\text{PRED}\ n))))$

▶  $\text{FACTORIAL} = Y\ H$

$H$  takes a function  $f$  and a number  $n$ . It returns 1 if the number is 0, oth

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FACTORIAL 5(OVERVIEW)

- ▶  $H = \lambda fn.(ISZERO\ n\ 1\ (\text{MULT}\ n\ (f\ (\text{PRED}\ n))))$

▶  $\text{FACTORIAL} = Y\ H$

$H$  takes a function  $f$  and a number  $n$ . It returns 1 if the number is 0, oth

<https://eduassistpro.github.io>

$$= H\ (Y\ H)\ 5\ \quad r!)$$

Add WeChat edu\_assist\_pro

## FACTORIAL 5(OVERVIEW)

►  $H = \lambda fn.(ISZERO\ n\ 1\ (\text{MULT}\ n\ (f\ (\text{PRED}\ n))))$

►  $\text{FACTORIAL} = Y\ H$

$H$  takes a function  $f$  and a number  $n$ . It returns 1 if the number is 0, oth

<https://eduassistpro.github.io>

$$= H\ (Y\ H)\ 5 \quad r!)$$

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

## FACTORIAL 5(OVERVIEW)

- ▶  $H = \lambda fn.(ISZERO n) 1 (MULT n (f (PRED n)))$

►  $FACTORIAL = Y H$

$H$  takes a function  $f$  and a number  $n$ . It returns 1 if the number is 0, oth

<https://eduassistpro.github.io>

$$= H (Y H) 5 \quad r!)$$

$$= 5 * ((Y H) 4)$$

$$= \dots$$

$$= 120 * ((Y H) 0)$$

Add WeChat `edu_assist_pro`

## FACTORIAL 5(OVERVIEW)

- $H = \lambda fn.(ISZERO n) 1 (MULT n (f (PRED n)))$

►  $FACTORIAL = Y H$

$H$  takes a function  $f$  and a number  $n$ . It returns 1 if the number is 0, oth

<https://eduassistpro.github.io>

$$= H (Y H) 5 \quad r!)$$

$$= 5 * ((Y H) 4)$$

$$= \dots$$

$$= 120 * ((Y H) 0)$$

$$= 120 * 1 = 120$$

Add WeChat `edu_assist_pro`

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

=

=<https://eduassistpro.github.io>

=

=Add WeChat edu\_assist\_pro

=

=

=

## FACTORIAL 3 (DETAILED 1)

Assignment Project Exam Help

=

=<https://eduassistpro.github.io>

=

=Add WeChat edu\_assist\_pro

=

=

=

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

= inator)

=  
<https://eduassistpro.github.io>

=

= Add WeChat edu\_assist\_pro

=

=

=

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

= inator)

= <https://eduassistpro.github.io?H>

=

= Add WeChat edu\_assist\_pro

=

=

=

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

= inator)

= <https://eduassistpro.github.io?H3> (β)

= Add WeChat edu\_assist\_pr

=

=

=

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

= inator)

= <https://eduassistpro.github.io>)<sup>2</sup> (H)  
= 3 (β)

= (ISZERO 3)  $\lambda$  (MULT 3 (Y H))<sup>2</sup> (β)  
= Add WeChat edu\_assist\_pr

=

=

=

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

= inator)

= <https://eduassistpro.github.io> )? (H)  
= 3 (β)

= (ISZERO 3) ↗ (MULT 3 (Y H)) (β)  
= Add WeChat edu\_assist\_pro / 0)

=

=

## FACTORIAL 3 (DETAILED 1)

# Assignment Project Exam Help

= inator)

= <https://eduassistpro.github.io> )? (H)  
= 3 (β)

= (ISZERO 3) ↗ (MULT 3 (Y H)) (β)

= ... = FALSE 1 (MULT 3 (Y H / 0))

= ... = MULT 3 (Y H (PRED 3)) (def. FALSE)

=

Add WeChat edu\_assist\_pro

## FACTORIAL 3 (DETAILED 1)

Assignment Project Exam Help

= inator)

= <https://eduassistpro.github.io>)<sup>2</sup> (H)  
= 3 (β)

= (ISZERO 3)  $\lambda$  (MULT 3 (Y H))<sup>(β)</sup>

= ... = FALSE 1 (MULT 3 (Y H / 0))

= ... = MULT 3 (Y H (PRED 3)) (def. FALSE)

= ... = MULT 3 (Y H 2) (PRED 3 = 2)

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

= ... = *MULT 3 (Y H 2)*

<https://eduassistpro.github.io>

=

=

Add WeChat edu\_assist\_pro

=

=

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

= ... = FACTORIAL 3 (Y H 2)

<https://eduassistpro.github.io>

=

=

Add WeChat edu\_assist\_pro

=

=

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

= ... = FACTORIAL<sup>3</sup>(Y H 2)

<https://eduassistpro.github.io>

=

=

Add WeChat edu\_assist\_pro

=

=

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

$$= \dots = \text{MULT } 3 (Y H 2)$$

<https://eduassistpro.github.io>

$$= \dots = \dots(\text{ISZERO } 0) 1\dots$$

Add WeChat edu\_assist\_pro

=

=

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

$$= \dots = \text{MULT } 3 (Y H 2)$$

<https://eduassistpro.github.io>

$$= \dots = \dots (\text{ISZERO } 0) 1 \dots$$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 ($$

Add WeChat edu\_assist\_pro

$$=$$
$$=$$

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

$$= \dots = \text{MULT } 3 (Y H 2)$$

<https://eduassistpro.github.io>

$$= \dots = \dots (\text{ISZERO } 0) 1 \dots$$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 ($$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 1)$$
$$=$$
$$=$$

Add WeChat `edu_assist_pro`

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

$$= \dots = \text{MULT } 3 (Y H 2)$$

<https://eduassistpro.github.io>

$$= \dots = \dots (\text{ISZERO } 0) 1 \dots$$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 ($$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 1)$$
$$= \dots = \text{MULT } 3 2$$
$$=$$

# Add WeChat edu\_assist\_pro

## FACTORIAL 3 (DETAILED 2)

# Assignment Project Exam Help

$$= \dots = \text{MULT } 3 (Y H 2)$$

<https://eduassistpro.github.io>

$$= \dots = \dots (\text{ISZERO } 0) 1 \dots$$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 ($$
$$= \dots = \text{MULT } 3 (\text{MULT } 2 1)$$
$$= \dots = \text{MULT } 3 2$$
$$= \dots = 6$$

Add WeChat `edu_assist_pro`

## Y COMBINATOR REMINDER

# Assignment Project Exam Help

This worked because the Y Combinator

Has the URL:  
<https://eduassistpro.github.io/>

Important:

- Add WeChat edu\_assist\_pro
- When performing the reductions, use that
  - *Don't*  $\beta$ -reduce the Y Combinator directly.

## OUTLINE

# Assignment Project Exam Help

- ▶ Revision - Lambda Calculus
- ▶ <https://eduassistpro.github.io/>
  - ▶ numbers (a different way)
  - ▶ pairs
  - ▶ lists
- ▶ Add WeChat edu\_assist\_pro
- ▶ Functional Programming

## BOOLEANS

# Assignment Project Exam Help

Reminder: our definition of Church Booleans lets us write:

simp <https://eduassistpro.github.io>

where  $B$  is some boolean, i.e. anything which re

- ▶  $\text{TRUE} = \lambda xy.x$ , or
- ▶  $\text{FALSE} = \lambda xy.y$

Add WeChat `edu_assist_pro`

## PAIRS (BARENDEGRT STYLE)

Let  $P, Q$  be expressions in the lambda calculus.

# Assignment Project Exam Help

If we write:

<https://eduassistpro.github.io>

Then:

Add WeChat edu\_assist\_pro

- ▶  $[M, N] \text{ TRUE} = M$
- ▶  $[M, N] \text{ FALSE} = N$

We can use  $[M, N]$  to denote an ordered pair.

## PAIRS (BARENDREGT)

[M, N] TRUE =  $(\lambda z.z\ M\ N) \text{ TRUE}$   
= TRUE M N

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## PAIRS (BARENDREGT)

[M, N] TRUE =  $(\lambda z.z\ M\ N)$  TRUE  
= TRUE M N

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pr`  
[M, N] FALSE =  $(\lambda z.z\ M)$   
= FALSE  
=  $(\lambda xy.y)\ M\ N$   
=  $(\lambda y.y)\ N$   
= N

## NUMBERS (BARENDEGRT STYLE)

Assignment Project Exam Help

Recall that predecessor, subtraction, division were difficult in the Church encoding.

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## NUMBERS (BARENDEGRT STYLE)

Assignment Project Exam Help

Recall that predecessor, subtraction, division were difficult in the Church encoding.

We can

- ▶ <https://eduassistpro.github.io>
- ▶  $n + 1 = [\text{FALSE}, n]$

Add WeChat edu\_assist\_pro

## NUMBERS (BARENDEGRT STYLE)

Assignment Project Exam Help

We can

- ▶ <https://eduassistpro.github.io>
- ▶  $n + 1 = [\text{FALSE}, n]$

For example:

- ▶  $1 = [\text{FALSE}, 0] = [\text{FALSE}, 1] = \dots$

## NUMBERS (BARENDEGRT STYLE)

Assignment Project Exam Help

We can

- ▶ <https://eduassistpro.github.io>
- ▶  $n + 1 = [\text{FALSE}, n]$

For example:

- ▶  $1 = [\text{FALSE}, 0] = [\text{FALSE}, I] =$
- ▶  $2 = [\text{FALSE}, 1] = [\text{FALSE}, [\text{FALSE}, I]]$

## NUMBERS (BARENDEGRT STYLE)

# Assignment Project Exam Help

We can

- ▶ <https://eduassistpro.github.io>
- ▶  $n + 1 = [\text{FALSE}, n]$

For example:

- ▶  $1 = [\text{FALSE}, 0] = [\text{FALSE}, I] =$
- ▶  $2 = [\text{FALSE}, 1] = [\text{FALSE}, [\text{FALSE}, I]]$
- ▶  $3 = [\text{FALSE}, 2] = [\text{FALSE}, [\text{FALSE}, [\text{FALSE}, I]]]$

## NUMBERS (BARENDEGRT STYLE)

Some of the operators are a *lot* simpler:

- ▶  $SUCC = \lambda x.[FALSE, x]$  (the next number)
  - ▶ This simply puts another FALSE in front.
  - ▶  $SUCC\ ONE = (\lambda x.[FALSE, x])\ ONE = [FALSE, ONE] =$

▶ <https://eduassistpro.github.io>

$$[FALSE, I] FALSE = I$$

▶  $ISZERO = \lambda x.x\ TRUE$

Add WeChat edu\_assist\_pro

## NUMBERS (BARENDEGRT STYLE)

Some of the operators are a *lot* simpler:

- ▶  $SUCC = \lambda x.[FALSE, x]$  (the next number)
  - ▶ This simply puts another FALSE in front.
  - ▶  $SUCC\ ONE = (\lambda x.[FALSE, x])\ ONE = [FALSE, ONE] =$

▶ <https://eduassistpro.github.io>

$$[FALSE, I]FALSE = I$$

▶  $ISZERO = \lambda x.x\ TRUE$

Add WeChat edu\_assist\_pro

... recall that PRED for the Church numerals was

$$\lambda nfx.n(\lambda gh.h(gf))(\lambda y.x)(\lambda u.u)$$

## NUMBERS (BARENDEGRT STYLE)

# Assignment Project Exam Help

Addition is more complex, but quite intuitive

- ▶ base case:  $ADD(0, y) = y$
- ▶

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## NUMBERS (BARENDEGRT STYLE)

# Assignment Project Exam Help

Addition is more complex, but quite intuitive

- ▶ base case:  $ADD(0, y) = y$
- ▶

<https://eduassistpro.github.io>

To im

$ADD = Y(\lambda fxy.ISZERO\ x)\ y\ (\lambda yf.f(y))$

- ▶ i.e.  $Y$  “if  $x$  is 0 then  $y$  else  $(1 + f(x - 1, y))$ ”

## GENERALISED RECURSION

# Assignment Project Exam Help

We can generalise this idea of recursion to support an arbitrary num

<https://eduassistpro.github.io>

See section 3.11 in the reference text (Barendregt) if you're interested in the fine details of this.

Add WeChat `edu_assist_pro`

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

=

=

=

=

=

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

=

=

=

=

=

=

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

=

=

=

=

=

=

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

$$= (\lambda xyz.$$

Add WeChat edu\_assist\_pro

=

=

=

=

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

$$= (\lambda xyz.$$

$$\equiv (\lambda yz.z$$

=

=

=

=

=

Add WeChat `edu_assist_pro`

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

$$= (\lambda xyz.$$

$$\equiv (\lambda y.z$$

$$= (\lambda z.za$$

=

=

=

=

Add WeChat `edu_assist_pro`

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

$$= (\lambda xyz.$$

$$\equiv (\lambda yz.z$$

$$= (\lambda z.za$$

$$= \text{TRUE } a \ b$$

$$=$$

$$=$$

$$=$$

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

$$= (\lambda xyz.$$

$$\equiv (\lambda yz.z$$

$$= (\lambda z.za$$

$$= \text{TRUE } a \ b$$

$$= (\lambda xy.x)ab$$

$$=$$

$$=$$

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

## PAIRS (CHURCH)

Similar idea, but not identical to the encoding Barendregt uses.

- ▶  $\text{PAIR} = \lambda xyz.zxy$
- ▶  $\text{FIRST} = \lambda p.p \text{ TRUE}$
- ▶  $\text{SECOND} = \lambda p.p \text{ FALSE}$

e.g.

<https://eduassistpro.github.io>

$$= (\lambda xyz.$$

$$\equiv (\lambda y.z$$

$$= (\lambda z.za$$

$$= \text{TRUE } a \ b$$

$$= (\lambda xy.x)ab$$

$$= (\lambda y.a)b$$

$$= a$$

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

## LIST (CHURCH)

# Assignment Project Exam Help

Idea: lists are pairs of (head, tail)



► <https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

I will denote lists as  $\{a, b, c, d, \dots\}$

## LIST (CHURCH)

We need a way to signal if the list is empty.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## LIST (CHURCH)

We need a way to signal if the list is empty.

Idea: each list entry is a nested pair (`isempty`, (`head`, `tail`))

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pro`

## LIST (CHURCH)

We need a way to signal if the list is empty.

Idea: each list entry is a nested pair (`isempty`, (`head`, `tail`))

# Assignment Project Exam Help

- Empty list =  $\text{NIL} = \text{PAIR } \text{TRUE } \text{ TRUE}$



- <https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## LIST (CHURCH)

We need a way to signal if the list is empty.

Idea: each list entry is a nested pair (`isempty`, (`head`, `tail`))

# Assignment Project Exam Help

► Empty list =  $\text{NIL} = \text{PAIR } \text{TRUE } \text{TRUE}$

►

►

<https://eduassistpro.github.io>

A list containing  $\{a, b, c, d\}$  would look like

Add WeChat `edu_assist_pro`

$(\text{PAIR } \text{FALSE } (\text{PAIR } a$   
 $\quad (\text{PAIR } \text{FALSE } (\text{PAIR } b$   
 $\quad (\text{PAIR } \text{FALSE } (\text{PAIR } c$   
 $\quad (\text{PAIR } \text{FALSE } (\text{PAIR } d \text{ NIL})))))))$

## LIST (CHURCH)

# Assignment Project Exam Help

To make our lists useful, we want the following functions:

- ▶
  - ▶ <https://eduassistpro.github.io>
  - ▶
  - ▶ *TAIL* gets the rest
  - ▶ *CONS* prepends a given value to the head of a list
- Add WeChat [edu\\_assist\\_pro](https://eduassistpro.github.io)

## LIST (CHURCH)

# Assignment Project Exam Help

Encoding:

- ▶
- ▶ <https://eduassistpro.github.io>
- ▶
- ▶  $TAIL =$
- ▶  $CONS =$
- ▶ Add WeChat edu\_assist\_pro

## LIST (CHURCH)

# Assignment Project Exam Help

Encoding:

- ▶
- ▶ <https://eduassistpro.github.io>
- ▶
- ▶  $TAIL =$
- ▶  $CONS =$
- ▶ Add WeChat edu\_assist\_pro

## LIST (CHURCH)

# Assignment Project Exam Help

Encoding:

- ▶
- ▶ <https://eduassistpro.github.io>
- ▶
- ▶  $TAIL =$
- ▶  $CONS =$
- ▶ Add WeChat edu\_assist\_pro

## LIST (CHURCH)

# Assignment Project Exam Help

Encoding:

- ▶
- ▶ <https://eduassistpro.github.io>
- ▶
- ▶  $TAIL =$
- ▶  $CONS =$
- ▶ Add WeChat edu\_assist\_pro

## LIST (CHURCH)

# Assignment Project Exam Help

Encoding:

- ▶
- ▶ <https://eduassistpro.github.io>
- ▶
- ▶  $TAIL = \lambda z. SECOND (SECOND z)$
- ▶ Add WeChat edu\_assist\_pro

## LIST (CHURCH)

# Assignment Project Exam Help

Encoding:

- ▶
  - ▶ <https://eduassistpro.github.io>
  - ▶
  - ▶  $TAIL = \lambda z. SECOND (SECOND z)$
  - ▶  $CONS = \lambda h t. PAIR h (TAIL t)$
- Add WeChat edu\_assist\_pro

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

=

Add = WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

=

Add = WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

=

Add = WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

=

Add = WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR TRUE TRUE TR

Add<sub>=</sub>WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR TRUE TRUE TR

Add =  $\lambda xyz. zxy$  TRUE T  
WeChat

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR TRUE TRUE TR

AddWeChatedu\_assist\_pro  
=  $(\lambda xyz. zxy) \text{TRUE} \text{ T}$   
= ... = TRUE TRUE TR  
=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR TRUE TRUE TR

AddWeChatedu\_assist\_pro  
=  $(\lambda xyz. zxy) \text{TRUE} \text{ T}$   
= ... = TRUE TRUE TR  
= ... = TRUE

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

*ISNL {a, b, c, d}*

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

*ISNL {a, b, c, d}*

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

*ISNL {a, b, c, d}*

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

*ISNL {a, b, c, d}*

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

*ISNIL {a, b, c, d}*

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR a {

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR a {

=  $(\lambda xyz.zxy)$  FALSE (PAIR

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR a {

=  $(\lambda xyz.zxy)$  FALSE (PAIR

= ... = TRUE FALSE (PAIR

=

Add WeChat edu\_assist\_pro

## LIST (CHURCH)

Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR a {

=  $(\lambda xyz.zxy)$  FALSE (PAIR

= ... = TRUE FALSE (PAIR

= ... = FALSE

Add WeChat edu\_assist\_pro

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

$\text{HEAD} \{a, b, c, d\}$

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

$\text{HEAD} \{a, b, c, d\}$

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

$\text{HEAD} \{a, b, c, d\}$

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

$\text{HEAD} \{a, b, c, d\}$

<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

$\text{HEAD } \{a, b, c, d\}$

<https://eduassistpro.github.io>

$= \text{SECOND } \{a, b, c, d\} \text{ TRU}$

Add WeChat edu\_assist\_pro

=

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

$\text{HEAD } \{a, b, c, d\}$

$= \text{SECOND } \{a, b, c, d\} \text{ TRU}$

$= (\lambda p. p \text{ FALSE}) \{a, b, c, d\}$

$=$

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= SECOND { $a, b, c, d$ } TRU

= ( $\lambda p. p$ ) FALSE { $a, b, c, d$ }  
= { $a, b, c, d$ } FALSE TRUE

=

Add WeChat edu\_assist\_pro

## LIST (CHURCH) EXAMPLE

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= SECOND { $a, b, c, d$ } TRU

= ( $\lambda p. p\ FALSE$ ) { $a, b, c, d$ }  
= { $a, b, c, d$ } FALSE TRUE

= PAIR FALSE (PAIR  $a$  { $b, c, d$ }) FALSE TRUE

= ...

Add WeChat edu\_assist\_pro

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help  
HEAD{a,b,c,d}  
...

<https://eduassistpro.github.io>

=

= Add WeChat edu\_assist\_pro

=

=

=

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\underline{= \dots}^{\text{HEAD } \{a, b, c, d\}}$

$E$   
<https://eduassistpro.github.io>

=

Add WeChat edu\_assist\_pro

=

=

=

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\underline{= \dots}^E$

$\underline{\text{https://eduassistpro.github.io}}$

=

= Add WeChat edu\_assist\_pro

=

=

=

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\underline{= \dots}^E$

$\underline{\text{https://eduassistpro.github.io}}$

=

= Add WeChat edu\_assist\_pro

=

=

=

## LIST (CHURCH) EXAMPLE (CONTINUED)

# Assignment Project Exam Help

E

<https://eduassistpro.github.io>

$= \text{FALSE } \text{FALSE } (\text{PAIR } a \{b,$

=Add WeChat edu\_assist\_pr

2

—

1

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\underline{=}^{\text{HEAD } \{a, b, c, d\}} \dots$

$E$

<https://eduassistpro.github.io>

$= \text{FALSE FALSE} (\text{PAIR } a \{b,$

$= \text{AddWeChat.edu_assist_pro} \text{PAIR } a \{b, c, d\} \text{ TRUE}$

$=$

$=$

$=$

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\underline{=}^{\text{HEAD } \{a, b, c, d\}} \dots E$   
<https://eduassistpro.github.io>

$= FALSE FALSE (\text{PAIR } a \{b,$

$= \dots \cdot \text{PAIR } a \{b, c, d\} \text{ TRUE}$

$= (\lambda xyz.zxy) a \{b, c, d\} \text{ TRUE}$

$=$

$=$

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\frac{HEAD\ \{a, b, c, d\}}{\underline{= \dots}}$

$E$

<https://eduassistpro.github.io>

$= FALSE\ FALSE\ (PAIR\ a\ \{b,$

$= \dots\ .\ .\ PAIR\ a\ \{b, c, d\}\ TRUE$

$= (\lambda xyz.zxy)\ a\ \{b, c, d\}\ TRUE$

$= \dots = TRUE\ a\ \{b, c, d\})$

(3  $\beta$ -reductions)

$=$

## LIST (CHURCH) EXAMPLE (CONTINUED)

Assignment Project Exam Help

$\underline{=}^{\text{HEAD } \{a, b, c, d\}} \dots E$   
<https://eduassistpro.github.io>

$= FALSE FALSE (\text{PAIR } a \{b,$

$= \dots \cdot \text{PAIR } a \{b, c, d\} \text{ TRUE}$

$= (\lambda xyz.zxy) a \{b, c, d\} \text{ TRUE}$

$= \dots = \text{TRUE } a \{b, c, d\})$

(3  $\beta$ -reductions)

$= \dots = a$

( $\text{TRUE } a \ b = a$ )

LIST (CHURCH) CONS

# Assignment Project Exam Help

$CONS = \lambda ht.PAIR\ FALSE\ (PAIR\ h\ t)$

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

LIST (CHURCH) CONS

# Assignment Project Exam Help

$CONS = \lambda ht.PAIR\ FALSE\ (PAIR\ h\ t)$

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

LIST (CHURCH) CONS

# Assignment Project Exam Help

$CONS = \lambda ht.PAIR\ FALSE\ (PAIR\ h\ t)$

<https://eduassistpro.github.io>

$= (\lambda t.PAIR\ FALSE\ (PA$

Add WeChat edu\_assist\_pro

=

LIST (CHURCH) CONS

# Assignment Project Exam Help

$$\text{CONS} = \lambda ht.\text{PAIR } \text{FALSE } (\text{PAIR } h\ t)$$

<https://eduassistpro.github.io>

$$= (\lambda t.\text{PAIR } \text{FALSE } (\text{PA}$$

$$= \text{PAIR } \text{FALSE } (\text{PAIR } a\ b)$$

=

Add WeChat edu\_assist\_pro

LIST (CHURCH) CONS

# Assignment Project Exam Help

$$\text{CONS} = \lambda ht.\text{PAIR } \text{FALSE } (\text{PAIR } h\ t)$$

<https://eduassistpro.github.io>

$$= (\lambda t.\text{PAIR } \text{FALSE } (\text{PA}$$

$$= \text{PAIR } \text{FALSE } (\text{PAIR } a\ N)$$

$$= \{a\}$$

Add WeChat edu\_assist\_pro

LIST (CHURCH) CONS

# Assignment Project Exam Help

<https://eduassistpro.github.io>

=  
= Add WeChat edu\_assist\_pro  
=  
=

LIST (CHURCH) CONS

# Assignment Project Exam Help

<https://eduassistpro.github.io>

=  
= Add WeChat edu\_assist\_pro  
=  
=

LIST (CHURCH) CONS

# Assignment Project Exam Help

$\text{CONS} := \lambda h. \lambda t. \text{PAIR} (\text{FALSE}, (\text{PAIR} h, t))$

=  
= Add WeChat edu\_assist\_pro  
=  
=

LIST (CHURCH) CONS

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR b (CONS

= Add WeChat edu\_assist\_pro

=

=

LIST (CHURCH) CONS

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR b (CONS

= PAIR FALSE (PAIR b (CONS

=

=

Add WeChat `edu_assist_pro`

LIST (CHURCH) CONS

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR b (CONS

= PAIR FALSE (PAIR b (CONS

= PAIR FALSE (PAIR b (PAIR )

=

Add WeChat **edu\_assist\_pro**

LIST (CHURCH) CONS

# Assignment Project Exam Help

<https://eduassistpro.github.io>

= PAIR FALSE (PAIR b (CONS

= PAIR FALSE (PAIR b (CONS

= PAIR FALSE (PAIR b (PAIR

= {b, a}

Add WeChat **edu\_assist\_pro**

)

## LIST ENCODINGS

# Assignment Project Exam Help

We now have structured data *and* recursion!

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## LIST ENCODINGS

# Assignment Project Exam Help

We now have structured data *and* recursion!

<https://eduassistpro.github.io>

Don't forget, just as there are many ways to represent lists in imperative programming, there are many possible representations of lists and other structures in lambda calculus.

Add WeChat `edu_assist_pro`

## OUTLINE

# Assignment Project Exam Help

- ▶ Revision - Lambda Calculus
- ▶ <https://eduassistpro.github.io/>
- ▶ numbers (a different way)
- ▶ pairs
- ▶ lists
- ▶ Add WeChat edu\_assist\_pro
- ▶ Functional Programming

# FIBONACCI

In the last tutorial, you probably implemented Fibonacci like this:

Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

# FIBONACCI

In the last tutorial, you probably implemented Fibonacci like this:

Assignment Project Exam Help

<https://eduassistpro.github.io>

This works, but is not very efficient (exponential time)

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

# FIBONACCI

In the last tutorial, you probably implemented Fibonacci like this:

Assignment Project Exam Help

<https://eduassistpro.github.io>

This works, but is not very efficient (exponential time).

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

In imperative programming you would use variable sequence (linear time complexity).

## FIBONACCI

In the last tutorial, you probably implemented Fibonacci like this:

Assignment Project Exam Help

<https://eduassistpro.github.io>

This works, but is not very efficient (exponential time).

Add WeChat `edu_assist_pro`

In imperative programming you would use variable sequence (linear time complexity).

A comparable approach in FP is to compute the sequence, e.g. as a list.

## LISTS IN LISP

# Assignment Project Exam Help

**nil** ; an empty list

**(cons** e l) ; add element e to list l

**(list** e) ; make a list containing element e

**(car** l) ; first element of list l

**(cdr** l) ; list l without its first element

**(last** l) ; the last element of list l

**(append** a b) ; combine two lists a and b

**(member** e l) ; first sublist starting at l which contains e

**(reverse** l) ; a mirror of the list l

Add WeChat edu\_assist\_pro

## LISTS IN LISP (EXAMPLES)

```
? (list 1 2 3)
(1 2 3)
? (cons 1 (cons 2 (cons 3 nil)))
(1 2 3)
```

```
? (
```

```
NIL
```

```
? (
```

```
(3 5)
```

```
? (cdr (list 1 2 3 4 5))
```

```
(2 3 4 5)
```

```
? (cdr (cdr (list 1 2 3 4 5)))
```

```
(3 4 5)
```

```
? (car (cdr (list 1 2 3 4 5)))
```

```
2
```

Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

## FIBONACCI

Idea: given part of the Fibonacci sequence and a number, add that many more elements of the sequence.

# Assignment Project Exam Help

```
(defun fib (n a)
  (if (
```

<https://eduassistpro.github.io>

```
    (cons
      (+ (car a) (car (cdr a)))
      a)
    ) ) ) )
```

```
(fib 100 (list 1 0))
```

Add WeChat edu\_assist\_pro

## FIBONACCI

Making it a bit nicer:

- We can make optional (default) arguments

- (`(car (cdr >)) = (cadr x)`)

- You can repeat the a, d as many times as required

(`def`

(`if`

(`fib (- n 1)`

(`ccons`

(`+ (car a) (cadr a))`

`a`

) ) ) )

(`fib 100`)

Add WeChat edu\_assist\_pro

## A NOTE ON LOOPS IN LISP

I avoided using loops to keep the first example closer to lambda calculus.

# Assignment Project Exam Help

There

<https://eduassistpro.github.io/>

```
(def (loop for f1 = 0 then f2
          and f2 = 1 then (+ f1 f2)
          repeat n finally r)
        (fib 100))
```

1

---

<sup>1</sup>source: <https://www.cliki.net/Fibonacci>

## JAVA

```
public boolean isPrime(long number) {
```

```
    return number > 1 &&  
        longStream  
            .rangeClosed(2, (long) Math.sqrt(number))
```

```
}
```

```
isPrim
```

<https://eduassistpro.github.io>

- ▶ “rangeClosed” gives a stream of values via `longStream`
- ▶ “noneMatch” checks the stream agains `isPrime`
- ▶ “variable -> expression” is a lambda abstraction!
  - ▶ It takes a value (index) from the range, and tests if it divides the number we’re checking.

---

<sup>2</sup>source: <https://www.voxxed.com/2015/12/functional-vs-imperative-programming-fibonacci-prime-and-factorial-in/>

JAVA

# Assignment Project Exam Help

```
public boolean isPrime(long number) {  
    return number > 1 &&  
        LongStream
```

<https://eduassistpro.github.io/>

```
}
```

```
isPrime(9220000000000000039L)
```

<sup>3</sup> Add WeChat edu\_assist\_pro

Adding ".parallel()" is enough magic sauce to get good speedup.

---

<sup>3</sup>source: <https://www.voxxed.com/2015/12/functional-vs-imperative-programming-fibonacci-prime-and-factorial-in/>

functional-vs-imperative-programming-fibonacci-prime-and-factorial-in-

## PYTHON

If you write much Python, you probably write more functional programming code than you thought.

&gt;

&gt;

5

&gt;

5

```
>>> [x + 5 for x in grades]
[48, 73, 40, 94, 72, 70, 75]
```

```
>>> max([x + 5 for x in g
```

94

Add WeChat `edu_assist_pro`

## PYTHON

```
>>> from functools import reduce
```

```
>>> prices = [43, 68, 35, 89, 67, 65, 70]
>>> sales = [3, 5, 0, 3, 2, 10, 30]
```

```
>
```

3 <https://eduassistpro.github.io>

- ▶ zip combines elements from two iterables i

▶ e.g. [(43, 3), (68, 5), ...]

## PYTHON

```
>>> from functools import reduce
```

```
>>> prices = [43, 68, 35, 89, 67, 65, 70]
```

```
>>> sales = [3, 5, 0, 3, 2, 10, 30]
```

```
>
```

3 <https://eduassistpro.github.io>

- ▶ zip combines elements from two iterables i.e. [(43, 3), (68, 5), ... ]
- ▶ map applies a function to every element of an it
  - ▶ e.g. [43\*3, 68\*5, ... ]

Add WeChat edu\_assist\_pro

## PYTHON

```
>>> from functools import reduce
```

```
>>> prices = [43, 68, 35, 89, 67, 65, 70]
```

```
>>> sales = [3, 5, 0, 3, 2, 10, 30]
```

```
>
```

3 <https://eduassistpro.github.io>

- ▶ zip combines elements from two iterables i.e. [(43, 3), (68, 5), ...]
- ▶ map applies a function to every element of an it
  - ▶ e.g. [43\*3, 68\*5, ... ]
- ▶ reduce combines the elements using a two parameter function
  - ▶ (((0+129) + 340) + 0) + ...

Add WeChat edu\_assist\_pro

## REVIEW

- ▶ Revision - Lambda Calculus
  - ▶ When  $\alpha$ -reductions are required
    - ▶  $\beta$ -reductions
  - ▶ Y Combinator

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- ▶ Encodings
  - ▶ numbers (in different ways)
  - ▶ pairs
  - ▶ lists
- ▶ Functional Programming
  - ▶ Using lists in LISP
  - ▶ Stream processing in Java
  - ▶ Some ubiquitous Python

Add WeChat edu\_assist\_pro