



程序代写代做 CS 编程辅导



MP4161



UNSW  
SYDNEY

## Advanced Topics in Software Verification

WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>

Gerwin Klein, June Andronick, Miki Tanaka, Johannes Åman Pohjola

T3/2022

# Binary Search (java.util.Arrays)

程序代写代做 CS编程辅导

```
1:  public static int binarySearch(int[] a, int key) {  
2:      int low = 0;  
3:      int high = a.length;  
4:  
5:      while (low <= high)  
6:          int mid = (low + high) / 2;  
7:          int midVal = a[mid];  
8:  
9:          if (midVal < key)  
10:             low = mid + 1;  
11:          else if (midVal > key)  
12:             high = mid - 1;  
13:          else  
14:              return mid; // key found.  
15:      }  
16:      return -(low + 1); // key not found.  
17:  }
```

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

6: QQ: 749389476  
int mid = (low + high) / 2;

<https://tutorcs.com>  
<http://googleresearch.blogspot.com/2006/06/extraread-all-about-it-nearly.html>



# Organisatorials

程序代写代做 CS编程辅导



WeChat: cstutorcs  
<http://www.cse.unsw.edu.au/~cs4161/>  
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# About us: Proofcraft and TS

程序代写代做 CS编程辅导

- **TS** (Trustworthy Systems) is a research group at UNSW
  - track record and real world impact in verified software
  - biggest achievement: formal verification of **seL4**
- **Proofcraft** is a new company
  - from former leaders of TS
  - providing services in software verification
- **seL4** is an operating microkernel used around the world in critical systems
  - with a proof of functional correctness and security:  
**Email: tutorcs@163.com**  
Security ↔ Isabelle/HOL model ↔ Haskell model ↔ C code  
↔ Binary   **QQ: 749389476**
  - 10 000 LOC / more than 1 million lines of proof
  - Open source, <https://tutorcs.com>

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

Security ↔ Isabelle/HOL model ↔ Haskell model ↔ C code

↔ Binary   **QQ: 749389476**

→ 10 000 LOC / more than 1 million lines of proof

→ Open source, [http://sel4.systems](https://sel4.systems)

We are always embarking on exciting new projects. Talk to us!

# What you will learn

程序代写代做 CS编程辅导

- how to use a theorem prover
- background, how to prove
- how to prove and reason
- how to reason about programs



WeChat: cstutorcs

Assignment Project Exam Help

**Health Warning**  
Email: tutorcs@163.com

**Theorem Proving is addictive**  
QQ: 749389476

<https://tutorcs.com>

# Prerequisites

程序代写代做 CS编程辅导

This is an advanced



It assumes knowledge in

- Functional programming
- First-order formal

The following program should make sense to you:

WeChat: cstutorcs

$$\begin{array}{rcl} \text{map } f [] & = & [] \\ \text{map } f (x:xs) & = & f x : \text{map } f xs \end{array}$$

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

You should be able to read and understand this formula:

QQ: 749389476

$$\exists x. (P(x) \rightarrow \forall x. P(x))$$

# Content — Using Theorem Provers

程序代写代做 CS编程辅导



Rough timeline

## → Foundations & Pi-Calculus

- Intro, Lambda calculus, natural deduction [1,2]
- Higher Order Logic, Pi-calculus (part 1) [2,3<sup>a</sup>]
- Term rewriting [3,4]

WeChat: cstutorcs

## → Proof & Specification Techniques

Assignment Project Exam Help

- Inductively defined sets, rule induction [4,5]
- Datatype induction, primitive recursion [5,7]
- General recursive functions, termination proofs [7<sup>b</sup>]
- Proof automation, Idris (part 2) [8]
- Hoare logic, proofs about programs, invariants [8,9]
- C verification [https://tutorcs.com] [9,10]
- Practice, questions, exam prep [10<sup>c</sup>]

Email: tutors@163.com

QQ: 749389476

https://tutorcs.com

<sup>a</sup>a1 due; <sup>b</sup>a2 due; <sup>c</sup>a3 due

# To have a chance at succeeding

程序代写代做 CS编程辅导

you should:

- attend lectures
- try Isabelle early
- redo all the demos
- try the exercises/homework we give, when we do give some  
**WeChat: cstutorcs**
- **DO NOT CHEAT**



- Assignments and exams are take-home. This does NOT mean you can work in groups. Each submission is personal.
- For more info, see Plagiarism Policy<sup>a</sup>

**QQ: 749389476**

**<https://tutorcs.com>**

---

<sup>a</sup> <https://student.unsw.edu.au/plagiarism>

# Credits

程序代写代做 CS编程辅导

some material (in us) from the m-provers part) shamelessly stolen  
from



WeChat: cstutorcs

Tobias Nipkow, Larry Paulson, Markus Wenzel

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>  
David Basin, Burkhardt Wolff

**Don't blame them, errors are ours**

# What is a formal proof?

程序代写代做 CS编程辅导

## A derivation in a formal calculus

Example:  $A \wedge B \vdash A \wedge B$  derivable in the following system

Rules:  $\frac{X \in S}{S \vdash X}$  (ass)       $\frac{S \cup \{X\} \vdash Y}{S \vdash X \rightarrow Y}$  (impl)

$$\frac{\begin{array}{c} S \vdash X \\ S \vdash Y \end{array}}{S \vdash X \wedge Y}$$
 (conjI)       $\frac{\begin{array}{c} S \cup \{X, Y\} \vdash Z \\ S \cup \{X \wedge Y\} \vdash Z \end{array}}{S \vdash X \wedge Y \vdash Z}$  (conjE)

Proof:

Assignment Project Exam Help

1.  $\{A, B\} \vdash B$  (by assumption)
2.  $\{A, B\} \vdash A$  (by assumption)
3.  $\{A, B\} \vdash B \wedge A$  (by conjI with 1 and 2)
4.  $\{A \wedge B\} \vdash B \wedge A$  (by conjE with 3)
5.  $\{\} \vdash A \wedge B \rightarrow B \wedge A$  (by impl with 4)

# What is a theorem prover?

程序代写代做 CS编程辅导

## Implementation of formal logic on a computer.

- fully automated (first order logic)
- automated, but not necessarily terminating (first order logic)
- with automation, interactive (higher order logic)
  
- based on rules and axioms
- can deliver proofs

WeChat: cstutorcs

Assignment Project Exam Help

There are other (algorithmic) verification tools:

Email: tutorcs@163.com

- model checking, static analysis, ...
- usually do not deliver proofs
- See COMP3153: Algorithmic Verification

QQ: 749389476  
<https://tutorcs.com>

# Why theorem proving?

程序代写代做 CS编程辅导

- Analysing systems thoroughly
- Finding design and implementation errors early
- High assurance (formal, machine checked proof)
- it's not always easy
- it's fun

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Main theorem proving system for this course

程序代写代做 CS编程辅导



Isabell

→ used here for applications, learning how to prove  
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# What is Isabelle?

程序代写代做 CS编程辅导

## A generic interactive proof assistant

→ **generic:**

not specialised to propositional or first-order logic  
(two large developments in HOL and ZF, will mainly use HOL)

→ **interactive:**

more than just yes/no, you can interactively guide the system

→ **proof assistant:**

Assignment Project Exam Help  
helps to explore, find, and maintain proofs

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>



# 程序代写代做 CS编程辅导



If I prove it on the computer, it is correct, right?

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

If I prove it on the computer, it is correct, right?

程序代写代做 CS编程辅导

No, because:

- ① hardware could be faulty
- ② operating system could be faulty
- ③ implementation runtime system could be faulty
- ④ compiler could be faulty
- ⑤ implementation could be faulty
- ⑥ logic could be inconsistent
- ⑦ theorem could mean something else



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

If I prove it on the computer, it is correct, right?

程序代写代做 CS编程辅导

No, but:

probability for



- OS and H/W issues by using different systems
- runtime/compiler bugs caused by using different compilers
- faulty implementation reduced by having the right prover architecture
- inconsistent logic reduced by implementing and analysing it
- wrong theorem reduced by expressive/intuitive logics

WeChat: cstutorcs  
Assignment Project Exam Help

Email: tutorcs@163.com

No guarantees, but assurance immensely higher than manual

proof  
<https://tutorcs.com>

If I prove it on the computer, it is correct, right?

程序代写代做 CS编程辅导

Soundness architecture

careful implementation



LCF approach, small proof kernel

WeChat: cstutorcs

explicit proofs + proof checker

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

PVS

ACL2

HOL4

Isabelle

HOL-light

Assignment Project Exam Help

Coq

Lean

Twelf

Isabelle

HOL4

Agda

# Meta Logic

程序代写代做 CS编程辅导

## Meta language:

The language used to talk about another language.



## Examples:

English in a Spanish class, English in an English class

## Meta logic:

WeChat: cstutorcs

The logic used to formalize another logic

Assignment Project Exam Help

## Example:

Mathematics used to formalize derivations in formal logic

QQ: 749389476

<https://tutorcs.com>

# Meta Logic – Example

程序代写代做 CS编程辅导

## Syntax:

Formulae:  $F ::= \text{True} \mid F \wedge F \mid \text{False}$   
 $V ::= \dots$



Derivable:  $S \vdash X$   $\lambda$  a formula,  $S$  a set of formulae

WeChat: cstutorcs

logic / meta logic  
Assignment Project Exam Help

$$\frac{X \in S}{S \models X} \quad \frac{S \cup \{X\} \vdash Y}{S \models X \rightarrow Y}$$

$$\frac{\begin{array}{c} S \vdash X \\ S \vdash Y \end{array}}{S \vdash X \wedge Y} \quad \frac{\begin{array}{c} S \cup \{X, Y\} \vdash Z \\ S \cup \{X \wedge Y\} \vdash Z \end{array}}{S \vdash X \wedge Y \vdash Z}$$

QQ: 749389476  
<https://tutorcs.com>

# Isabelle's Meta Logic

程序代写代做 CS编程辅导



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

$\wedge$

## 程序代写代做 CS编程辅导

**Syntax:**  $\lambda x. F$  another meta level formula)  
in ASCII:  $!x. F$



- universal quantifier on the meta level
- used to denote parameters
- example and more later

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

$\implies$

程序代写代做 CS编程辅导

**Syntax:**  $A \implies B$  ( $A, B$  other meta level formulae)

in ASCII: A ==> B



**Binds to the right:**

$$A \implies B \implies C = A \implies (B \implies C)$$

WeChat: cstutorcs

**Abbreviation:**

Assignment Project Exam Help

$$[A; B] \implies C = A \implies B \implies C$$

QQ: 749389476

→ read:  $A$  and  $B$  implies  $C$

→ used to write down rules, theorems, and proof states

<https://tutorcs.com>

## Example: a theorem

程序代写代做 CS编程辅导

**mathematics:** if  $x < 0$  and  $y < 0$ , then  $x + y < 0$



**formal logic:**  $\vdash \boxed{x < 0 \wedge y < 0} \rightarrow x + y < 0$

**variation:**  $x < \boxed{0} \vdash x + y < 0$

**Isabelle:**

variation:

variation:

WeChat: cstutorcs  
lemma “ $x < 0 \wedge y < 0 \rightarrow x + y < 0$ ”

Assignment Project Exam Help  
lemma “ $[x < 0; y < 0] \Rightarrow x + y < 0$ ”

lemma

assumes “ $x < 0$ ” and “ $y < 0$ ” shows “ $x + y < 0$ ”

QQ: 749389476

<https://tutorcs.com>

## Example: a rule

程序代写代做 CS编程辅导

**logic:**



$$S \vdash X \quad S \vdash Y$$

WeChat: cstutorcs

**variation:**

**Isabelle:**

Assignment Project Exam Help  
 $[X; Y] \Rightarrow X \wedge Y$

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

## Example: a rule with nested implication

程序代写代做 CS编程辅导

logic:



variation:

$$\frac{S \cup \{X\} \vdash Z \quad S \cup \{Y\} \vdash Z}{\text{Assignment Project Exam Help}}$$

Isabelle:

$$\frac{[X \vee Y; X \Rightarrow Z; Y \Rightarrow Z] \Rightarrow Z}{\text{QQ: 749389476}}$$

<https://tutorcs.com>

$\lambda$

## 程序代写代做 CS编程辅导

**Syntax:**  $\lambda x. F$

in ASCII: %x. F



another meta level formula)

→ lambda abstraction

→ used for functions in object logics

→ used to encode bound variables in object logics

→ more about this soon

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



# Enough Theory!

Assignment Project Exam Help

Getting started with Isabelle

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# System Architecture

程序代写代做 CS编程辅导

Prover IDE (jEdit interface)



HOL, ZF – others

Isabelle – generic, interactive theorem prover

WeChat: cstutorcs  
Standard ML – logic implemented as ADT

Assignment Project Exam Help  
User can access all layers!

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# System Requirements

程序代写代做 CS编程辅导

→ Linux, Windows



OS X (10.8 +)

→ Standard ML (P



llementation)

→ Java (for jEdit)



WeChat: cstutorcs

Premade packages for Linux, Mac, and Windows + info on:

Assignment Project Exam Help

<http://mirror.cse.unsw.edu.au/pub/isabelle/>

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Documentation

程序代写代做 CS编程辅导

Available from <http://isabelle.in.tum.de>

→ Learning Isabelle



- Concrete Semantics Book
- Tutorial on Isabelle/Isar (LNCS 2283)
- Tutorial on Isar
- Tutorial on Locales

WeChat: cstutorcs

→ Reference Manuals

Assignment Project Exam Help

- Isabelle/Isar Reference Manual
- Isabelle Reference Manual
- Isabelle System Manual

Email: tutorcs@163.com

→ Reference Manuals for Object Logics

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



Demo

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# jEdit/PIDE

程序代写代做 CS编程辅导



WeChat: cstutorcs  
Assignment Project Exam Help  
Email: tutorcs@163.com  
QQ: 749389476  
<https://tutorcs.com>

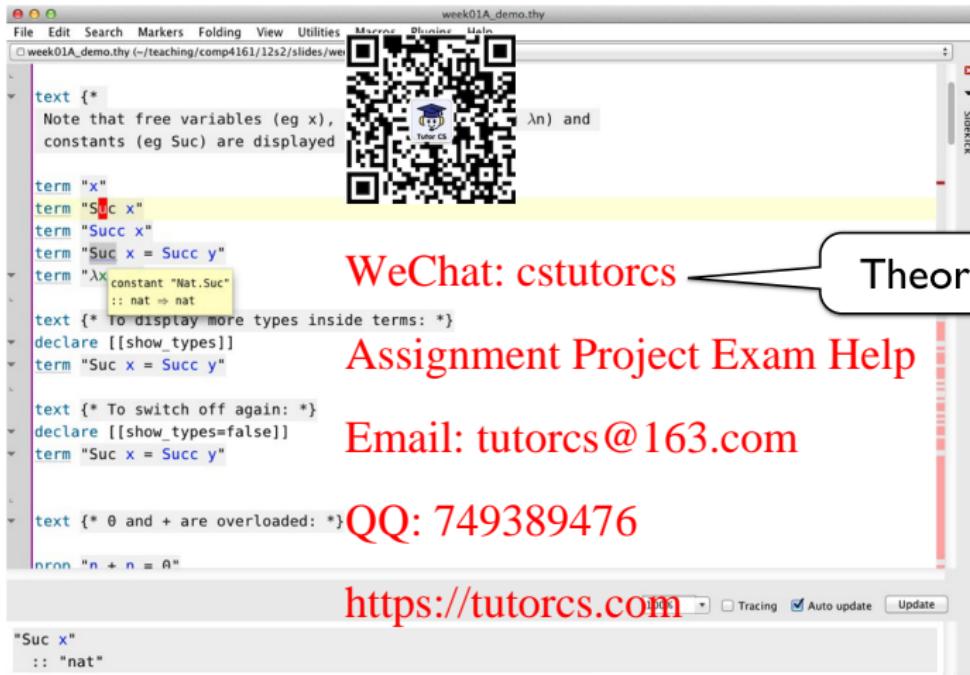
```
week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Main
week01A_demo.thy (~/teaching/comp4161/12s2/slides/we
text {* Note that free variables (eg x), constants (eg Suc) are displayed
term "x"
term "Suc x"
term "Succ x"
term "Suc x = Succ y"
term "\lambda x constant "Nat.Suc"
      :: nat -> nat
text {* To display more types inside terms: *}
declare [[show_types]]
term "Suc x = Succ y"

text {* To switch off again: *}
declare [[show_types=false]]
term "Suc x = Succ y"

text {* 0 and + are overloaded: *}
declare "n + n = A"
"Suc x"
      :: "nat"
```

# jEdit/PIDE

程序代写代做 CS编程辅导



```
week01A_demo.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
week01A_demo.thy (~/teaching/comp4161/12s2/slides/we
text {* Note that free variables (eg x), constants (eg Suc) are displayed
term "x"
term "Suc x"
term "Succ x"
term "Suc x = Succ y"
term "\lambda x constant "Nat.Suc"
    :: nat -> nat
text {* To display more types inside terms: *}
declare [[show_types]]
term "Suc x = Succ y"

text {* To switch off again: *}
declare [[show_types=false]]
term "Suc x = Succ y"

text {* 0 and + are overloaded: *}
operator "+ n = A"
"Suc x"
    :: "nat"
```

A QR code is displayed in the center of the window, with the text "TutorCS" and a graduation cap icon below it.

WeChat: cstutorcs

Theory File

Assignment Project Exam Help

Email: tutorcs@163.com

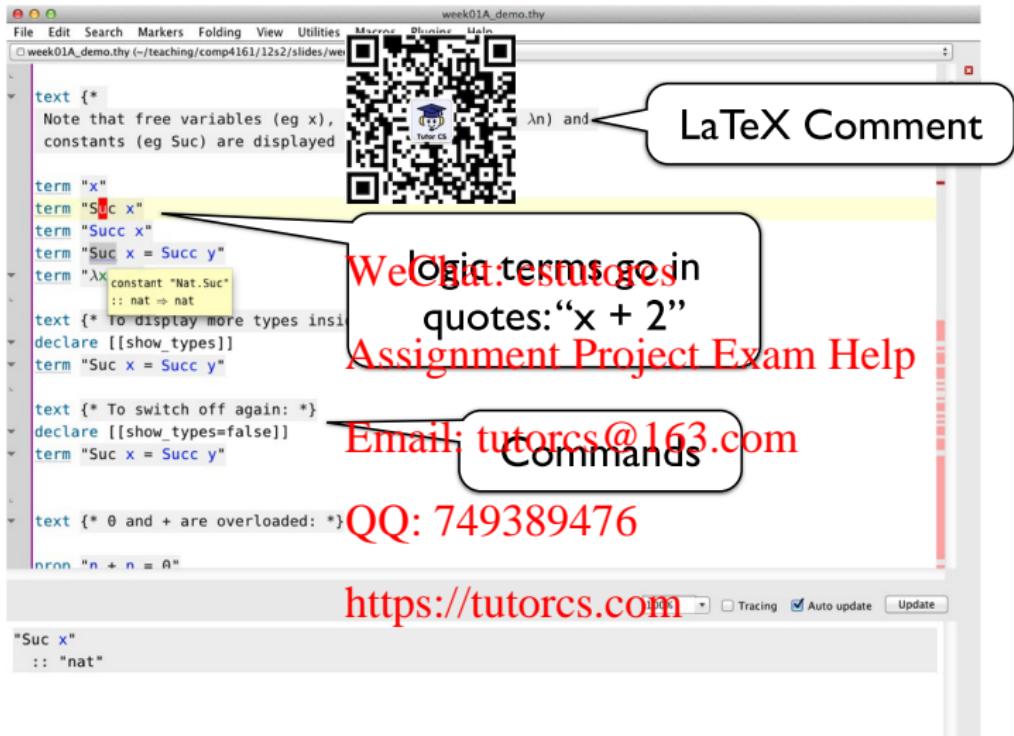
QQ: 749389476

<https://tutorcs.com>

Isabelle Output

# jEdit/PIDE

程序代写代做 CS编程辅导



The screenshot shows the jEdit/PIDE interface with a file named "week01A\_demo.thy". A yellow box highlights a LaTeX comment: "Note that free variables (eg x), constants (eg Suc) are displayed". A callout bubble points to this with the text "LaTeX Comment". Another yellow box highlights the term "Suc x", which is also mentioned in the LaTeX comment. A callout bubble points to this with the text "We often terms go in quotes: "x + 2"".

Assignment Project Exam Help

Email: tutorcs@163.com

Commands

QQ: 749389476

<https://tutorcs.com>

"Suc x"  
:: "nat"

# jEdit/PIDE

## 程序代写代做 CS编程辅导



week01A\_demo.thy

```
text {*  
Note that free variables (eg x),  
constants (eg Suc) are displayed  
  
term "x"  
term "Suc x"  
term "Succ x"  
term "Suc x = Succ y"  
term "\lambda x constant "Nat.Suc"  
:: nat => nat  
text {* To display more types inside terms: *}  
declare [[show_types]]  
term "Suc x = Succ y"  
  
text {* To switch off again: *}  
declare [[show_types=false]]  
term "Suc x = Succ y"  
  
text {* 0 and + are overloaded: *}  
operator "+ n = A"  
operator "n + 0 = A"  
  
"Suc x"  
:: "nat"
```

WeChat: csCommand click  
jumps to definition  
Assignment Project Exam Help

Email: tutorcs@163.com  
Command + hover  
for popup info  
QQ: 749389476

<https://tutorcs.com>

# jEdit/PIDE

程序代写代做 CS编程辅导

The screenshot shows the jEdit/PIDE interface with a code editor displaying a file named `week01A_demo.thy`. The code is a Coq script containing various terms and declarations. Several parts of the code are highlighted in yellow, and a QR code is displayed above the editor area.

Annotations overlaid on the image include:

- A large yellow speech bubble containing the text "processed". It points to the right side of the code editor.
- A smaller yellow speech bubble containing the text "error". It points to the right side of the code editor.
- A red speech bubble containing the text "unprocessed". It points to the right side of the code editor.
- Red text overlays:
  - WeChat: cstutorcs
  - Email: tutorcs@163.com
  - QQ: 749389476
  - Assignment Project Exam Help
  - <https://tutorcs.com>

## Exercises

程序代写代做 CS编程辅导



- Download and install Isabelle from  
<http://mirror.cse.unsw.edu.au/pub/isabelle/>
- Step through the demo files from the lecture web page
- Write your own theory file, look at some theorems in the library, try 'find\_theorems'
- How many theorems can help you if you need to prove something containing the term "Suc(Suc x)"?
- What is the name of the theorem for associativity of addition of natural numbers in the library?

WeChat: cstutorcs

Assignment Project Exam Help

Email: [tutors@163.com](mailto:tutors@163.com)

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



$\lambda$ -Calculus  
WeChat: cstutorcs  
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Content

## 程序代写代做 CS编程辅导

### → Foundations & Principles

- Intro, Lambda calculus, natural deduction [1,2]
- Higher Order Logic [2,3<sup>a</sup>]
- Term rewriting [3,4]



### → Proof & Specification Techniques

- Inductively defined sets, rule induction [4,5]
- Datatype induction, primitive recursion [5,7]
- General recursive functions, termination proofs [7<sup>b</sup>]
- Proof automation [8]
- Hoare logic, proofs about programs, invariants [8,9]
- C verification [9,10]
- Practice, questions, exam prep [10<sup>c</sup>]

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

---

<sup>a</sup>a1 due; <sup>b</sup>a2 due; <sup>c</sup>a3 due

# $\lambda$ -calculus

程序代写代做 CS编程辅导

## Alonzo Church

- lived 1903–1995
- supervised people
- famous for Church-Turing thesis, lambda calculus, first undecidability results
- invented  $\lambda$  calculus in 1930's



Turing, Stephen Kleene



WeChat: cstutorcs

## $\lambda$ -calculus

Assignment Project Exam Help

- originally meant as foundation of mathematics
- important applications in theoretical computer science
- foundation of computability and functional programming

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# untyped $\lambda$ -calculus

程序代写代做 CS编程辅导

- turing complete n
- a simple way of w



Basic intuition:

instead of  $f(x) = x + 5$   
write  $\lambda x. x + 5$

$\lambda x. x + 5$

Assignment Project Exam Help

- a term
- a nameless function
- that adds 5 to its parameter

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Function Application

程序代写代做 CS编程辅导

For applying arguments to functions



and of  $f(a)$   
 $f\ a$

WeChat: cstutorcs

**Example:**  $(\lambda x. x + 5) \ a$   
Assignment Project Exam Help

**Evaluating:** in  $(\lambda x. t) \ a$  replace  $x$  by  $a$  in  $t$   
(computation!) QQ: 749389476

**Example:**  $(\lambda x. x + 5) \ (a + b)$  evaluates to  $(a + b) + 5$   
<https://tutorcs.com>

程序代写代做 CS编程辅导



That's it!  
WeChat: cstutorcs  
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



Now Formal  
WeChat: cstutorcs  
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Syntax

程序代写代做 CS编程辅导

**Terms:**  $t$  |  |  $c$  |  $(t\ t)$  |  $(\lambda x. t)$   
 $v, \nu$ , ,  $c \in C$ ,  $V, C$  sets of names

WeChat: cstutorcs

- $\lambda x. x$  variables
  - $c$  constants
  - $(t\ t)$  application
  - $(\lambda x. t)$  abstraction
- QQ: 749389476

<https://tutorcs.com>

# Conventions

程序代写代做 CS编程辅导

- leave out parentheses where possible
- list variables instead of numbers: example  $\lambda$



**Example:** instead of  $(\lambda x. (\lambda y. (x y)))$  write  $\lambda y. x. x y$

## Rules:

WeChat: cstutorcs

- list variables:  $\lambda x. (\lambda y. t) = \lambda x. y. t$
- application binds to the left:  $x y z = (x y) z \neq x(y z)$
- abstraction binds to the right:  $\lambda x. x y = \lambda x. (x y) \neq (\lambda x. x) y$
- leave out outermost parentheses

QQ: 749389476

<https://tutorcs.com>

# Getting used to the Syntax

程序代写代做 CS编程辅导

**Example:**

$$\lambda x \ y \ z. \ x \ z \ (y \ z) =$$



$$\lambda x \ y \ z. \ (x \ z) \ (y \ z)$$

$$\lambda x \ y \ z. \ ((x \ z) \ (y \ z))$$

WeChat: cstutorcs

$$\lambda x. \ \lambda y. \ \lambda z. \ ((x \ z) \ (y \ z)) =$$

Assignment Project Exam Help

$$(\lambda x. \ (\lambda y. \ (\lambda z. \ ((x \ z) \ (y \ z))))))$$

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Computation

程序代写代做 CS编程辅导

**Intuition:** replace parameter by argument  
this is called substitution



**Remember:**  $(\lambda x. t) a \rightarrow_{\beta} t$  evaluated (noted  $\rightarrow_{\beta}$ ) to  $t$  where  $x$  is replaced by  $a$

**Example**

WeChat: cstutorcs

$(\lambda x. y. Suc\ x = y) 3 \rightarrow_{\beta}$  Assignment Project Exam Help

$(\lambda x. (\lambda y. Suc\ x = y)) 3 \rightarrow_{\beta}$  Email: tutorcs@163.com

$(\lambda y. Suc\ 3 = y)$

QQ: 749389476

$(\lambda x y. f\ (y\ x)) 5 (\lambda x. x) \rightarrow_{\beta}$

$(\lambda y. f\ (y\ 5)) (\lambda x. x) \rightarrow_{\beta}$  https://tutorcs.com

$f\ ((\lambda x. x)\ 5) \rightarrow_{\beta}$

$f\ 5$

# Defining Computation

程序代写代做 CS编程辅导

$\beta$  reduction:

$$\begin{array}{lll} s \xrightarrow{\beta} s' & t \xrightarrow{\beta} t' & s) t \xrightarrow{\beta} s[t] \\ t \xrightarrow{\beta} t' & \Rightarrow & (s t) \xrightarrow{\beta} (s' t') \\ s \xrightarrow{\beta} s' & \Rightarrow \text{WeChat: (tutorcs)} & (\lambda x. s') \end{array}$$

Assignment Project Exam Help

Still to do: define  $s[x \leftarrow t]$   
Email: [tutorcs@163.com](mailto:tutorcs@163.com)

QQ: 749389476

<https://tutorcs.com>



# Defining Substitution

程序代写代做 CS编程辅导



Easy concept problem: variable capture.  
 $(\lambda x. x z)[z \leftarrow x]$

We do **not** want:  $(\lambda x. x x)$  as result.

Assignment Project Exam Help  
What do we want?

Email: [tutorcs@163.com](mailto:tutorcs@163.com)

In  $(\lambda y. y z)[z \leftarrow x] = (\lambda y. y x)$  there would be no problem.  
QQ: 749389476

So, solution is: rename bound variables.  
<https://tutorcs.com>

# Free Variables

程序代写代做 CS编程辅导

**Bound variables:** in  $(\lambda x. t)$ ,  $x$  is a bound variable.

**Free variables**  $FV$



$$FV(x) = \{\}$$

$$FV(c) = \{\}$$

$$FV(s t) = FV(s) \cup FV(t)$$

$$FV(\lambda x. t) = FV(t) \setminus \{x\}$$

WeChat: WeChat: **tutorcs**  
Assignment Project Exam Help

**Example:**  $FV(\lambda x. (x \vee (\lambda y. (y \wedge x))) \wedge x)$  = { $y$ }

Term  $\lambda y. (y \wedge x)$  is called **Closed** if  $FV(t) = \{\}$

The substitution example,  $(x \vee z)[z \leftarrow x]$ , is problematic because the bound variable  $x$  is a free variable of the replacement term “ $x$ ”.

# Substitution

程序代写代做 CS编程辅导

$$\begin{array}{lll} x [x \leftarrow t] & = & \text{QR code} \\ y [x \leftarrow t] & = & \text{QR code} \quad \text{if } x \neq y \\ c [x \leftarrow t] & = & \text{QR code} \end{array}$$

$$(s_1 s_2) [x \leftarrow t] = (s_1[x \leftarrow t] s_2[x \leftarrow t])$$

WeChat: cstutorcs

$$(\lambda x. s) [x \leftarrow t] = (\lambda x. s)$$

Assignment Project Exam Help

$$(\lambda y. s) [x \leftarrow t] = (\lambda y. s[x \leftarrow t])$$

if  $x \neq y$  and  $y \notin FV(t)$

$$(\lambda y. s) [x \leftarrow t] = (\lambda z. s[y \leftarrow z][x \leftarrow t])$$

if  $x \neq y$

and  $z \notin FV(t) \cup FV(s)$

QQ: 749389476

<https://tutorcs.com>

# Substitution Example

程序代写代做 CS编程辅导

$$\begin{aligned} & (x \ ( \lambda x. \ x) \ [x \leftarrow y]) [x \leftarrow y] \\ = & (x [x \leftarrow y] \ [x \leftarrow y]) [x \leftarrow y] \ ((\lambda y. \ z \ x) [x \leftarrow y]) \\ = & y \ (\lambda x. \ x) \end{aligned}$$

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

## $\alpha$ Conversion

程序代写代做 CS编程辅导

Bound names are irrelevant:

$\lambda x. x$  and  $\lambda y. y$  denote the same function.

$\alpha$  conversion:

$s =_{\alpha} t$  means  $s = t$  ignoring of bound variables.

Formally:

WeChat: cstutorcs

Assignment Project Exam Help

$$s \xrightarrow{\alpha} s' \implies (s t) \xrightarrow{\alpha} (s' t)$$

$$t \xrightarrow{\alpha} t' \implies (s t) \xrightarrow{\alpha} (s t')$$

$$s \xrightarrow{\alpha} s' \implies (\lambda x. s) \xrightarrow{\alpha} (\lambda x. s')$$

QQ: 749389476

<https://tutorcs.com>

$$s =_{\alpha} t \text{ iff } s \xrightarrow{\alpha}^* t$$

( $\xrightarrow{\alpha}^*$  = transitive, reflexive closure of  $\xrightarrow{\alpha}$  = multiple steps)

# $\alpha$ Conversion

程序代写代做 CS编程辅导

**Equality in Isa**  **quality modulo  $\alpha$  conversion:**

if  $s =_{\alpha} t$  then   $t$  are syntactically equal.

**Examples:**

WeChat: cstutorcs

$$x (\lambda x. y. x y)$$

$$=_\alpha x (\lambda y. x. y x) \text{ Assignment Project Exam Help}$$

$$=_\alpha x (\lambda z. y. z y)$$

$$\neq_\alpha z (\lambda z. y. z y) \text{ Email: tutorcs@163.com}$$

$$\neq_\alpha x (\lambda x. x. x x) \text{ QQ: 749389476}$$

<https://tutorcs.com>

Back to  $\beta$

## 程序代写代做 CS编程辅导

We have defined  $\beta$  reduction:  $s \rightarrow_\beta t$

Some notation and c



$\exists n. s \rightarrow_\beta^* n \wedge t \rightarrow_\beta^* n$

→  $t$  is **reducible** if there is an  $s$  such that  $t \rightarrow_\beta s$

→  $(\lambda x. s)$   $t$  is called a **redex** (reducible expression)

→  $t$  is reducible iff it contains a redex

→ if it is not reducible,  $t$  is in **normal form**

WeChat: cstutorcs

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# Does every $\lambda$ term have a normal form?

程序代写代做 CS编程辅导



No!

Example:

$$\begin{aligned} & (\lambda x. x x) (\lambda x. x x) \xrightarrow{\beta} \\ & (\lambda x. x x) (\lambda x. x x) \xrightarrow{\beta} \\ & (\lambda x. x x) (\lambda x. x x) \xrightarrow{\beta} \dots \end{aligned}$$

(but:  $(\lambda x. y. y) ((\lambda x. x x) (\lambda x. x x)) \xrightarrow{\beta} \lambda y. y$ )

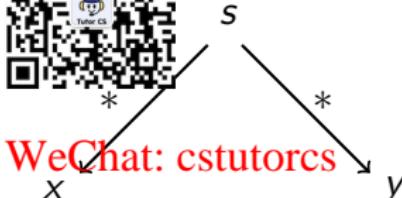
QQ: 749389476

$\lambda$  calculus is not terminating

# $\beta$ reduction is confluent

程序代写代做 CS编程辅导

**Confluence:**  $s \xrightarrow{*} \text{QR code} \xrightarrow{*_{\beta}} y \implies \exists t. x \xrightarrow{*_{\beta}} t \wedge y \xrightarrow{*_{\beta}} t$



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

Order of reduction does not matter for result  
<https://tutorcs.com>  
Normal forms in  $\lambda$  calculus are unique

# $\beta$ reduction is confluent

程序代写代做 CS编程辅导

Example:

$$(\lambda x. y. y) ((\lambda x. x x) \text{ QR code } \lambda x. y. y) (a a) \xrightarrow{\beta} \lambda y. y$$
$$(\lambda x. y. y) ((\lambda x. x x) \text{ QR code } \beta \lambda y. y)$$

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# $\eta$ Conversion

程序代写代做 CS编程辅导

Another case of trivially equal functions:  $t = (\lambda x. t x)$

Definition:



$$\begin{array}{lll} s \xrightarrow{\eta} s' & \Rightarrow & \xrightarrow{\eta} t \quad \text{if } x \notin FV(t) \\ t \xrightarrow{\eta} t' & \Rightarrow & \xrightarrow{\eta} (s \ t) \quad \xrightarrow{\eta} (s \ t') \\ s \xrightarrow{\eta} s' & \Rightarrow & \text{WeChat: cstutorcs} \quad \xrightarrow{\eta} (\lambda x. s') \end{array}$$

$$s =_{\eta} t \text{ iff } \exists n. s \xrightarrow{n} t \wedge t \xrightarrow{n} s$$

Email: tutorcs@163.com

Example:  $(\lambda x. f x) (\lambda y. g y) \xrightarrow{\eta} (\lambda x. f x) g \xrightarrow{\eta} f g$

QQ: 749389476

- $\eta$  reduction is confluent and terminating.
- $\xrightarrow{\beta\eta}$  is confluent.
- $\xrightarrow{\beta\eta}$  means  $\xrightarrow{\beta}$  and  $\xrightarrow{\eta}$  steps are both allowed.
- Equality in Isabelle is also modulo  $\eta$  conversion.

In fact ...

程序代写代做 CS编程辅导



Equality in Isat modulo  $\alpha$ ,  $\beta$ , and  $\eta$  conversion.

We will see later why that is possible.

WeChat: cstutorcs  
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导



**Isabelle Demo**  
WeChat: cstutorcs  
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

# So, what can you do with $\lambda$ calculus?

程序代写代做 CS编程辅导

$\lambda$  calculus is very expressive. You can encode:

- logic, set theory
- turing machines, programs, etc.



## Examples:

$$\text{true} \equiv \lambda x. y. x \quad \text{if true } x y \xrightarrow[\beta]{}^* x$$

$$\text{false} \equiv \lambda x. y. y \quad \text{if false } x y \xrightarrow[\beta]{}^* y$$

$$\text{if } \equiv \lambda z. x. z x y$$

WeChat: cstutorcs  
Assignment Project Exam Help  
Email: tutorcs@163.com

Now, not, and, or, etc is easy:

$$\text{not } \equiv \lambda x. \text{ if } x \text{ false true}$$

$$\text{and } \equiv \lambda x. y. \text{ if } x y \text{ false }$$

$$\text{or } \equiv \lambda x. y. \text{ if } x \text{ true } y$$

QQ: 749389476  
<https://tutorcs.com>

## More Examples

程序代写代做 CS编程辅导

### Encoding natural numbers (Church Numerals)



$$0 \equiv \lambda f x. x$$

$$1 \equiv \lambda f x. f x$$

$$2 \equiv \lambda f x. f(f x)$$

$$3 \equiv \lambda f x. f(f(f x))$$

...

Assignment Project Exam Help

Numeral  $n$  takes arguments  $f$  and  $x$ , applies  $f$   $n$ -times to  $x$ .  
Email: [tutorcs@163.com](mailto:tutorcs@163.com)

$$\text{iszero} \equiv \lambda n. n(\lambda x. \text{false}) \text{true}$$

$$\text{succ} \equiv \lambda n f x. f(n f x)$$

$$\text{add} \equiv \lambda m n f x. m f x (n f x)$$

QQ: 749389476

QQ: 749389476

QQ: 749389476

# Fix Points

程序代写代做 CS编程辅导

$$(\lambda x f. f (x \times f)) \quad \text{QR code} \quad ((x \times f)) \quad t \rightarrow_{\beta}$$

$$(\lambda f. f ((\lambda x f. f ((x \times f) \times f))) \quad t \rightarrow_{\beta}$$
  
$$t \ ((\lambda x f. f (x \times f) \times f) \times f (x \times f)) \quad t)$$



$$\mu = (\lambda x f. f (x \times f))$$

$$\mu t \rightarrow_{\beta} t (\mu t) \xrightarrow{\text{WeChat: cstutorcs}} t (t (\mu t)) \rightarrow_{\beta} t (t (t (\mu t))) \rightarrow_{\beta} \dots$$

Assignment Project Exam Help

$(\lambda x f. f (x \times f))$  (~~Email: tutorcs@163.com~~ is Turing's fix point operator

QQ: 749389476

<https://tutorcs.com>

## Nice, but ...

程序代写代做 CS编程辅导

As a mathematical foundation,  $\lambda$  does not work. It resulted in an inconsistent logic.



- Frege (Predicate Logic, 1879):

allows arbitrary quantification over predicates

- Russell (1901): Paradox  $R \equiv \{X | X \notin X\}$

- Whitehead & Russell (Principia Mathematica, 1910-1913):

Fix the problem

- Church (1930):  $\lambda$  calculus as logic, true, false,  $\wedge$ , ... as  $\lambda$  terms

Assignment Project Exam Help

Email: tutorcs@163.com

### Problem:

with  $\{x | P(x)\} \equiv \forall x. P x \quad x \in M \equiv M x$

you can write  $R \equiv \lambda x. \text{not } (x x)$

and get  $(R R) =_{\beta} \text{not } (R R)$

because  $(R R) = (\lambda x. \text{not } (x x)) R \longrightarrow_{\beta} \text{not } (R R)$

# We have learned so far...

程序代写代做 CS编程辅导

- λ calculus syntax
- free variables, substitution
- β reduction
- α and η conversion
- β reduction is confluent
- λ calculus is very expressive (turing complete)
- λ calculus results in an inconsistent logic



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>