Assignment Project Exam Help

Software System Design and Implementation

https://powcoder.com

Add Write Edit 17 14 FCS TO WW Coder

Natural Deduction



Each connective typically has introduction and elimination rules. https://powcoder.com

Add WeChat powcoder

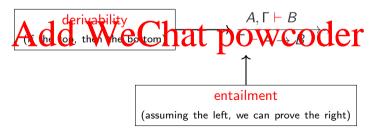
•0000

Typed Lambda Calculus

Natural Deduction

Logic We case it is that describe how to prove various connectives.

Each connective typically has *introduction* and *elimination* rules. For example, to professional pr assuming A. This introduction rule is written as:



More rules

Implication also has an elimination rule, that is also called *modus ponens*:

Assignment Project Exam Help

Conjunction (ald) the print of prior whether deriving in:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \land B} \land -1$$

 $\underset{\text{It has two elimination rules:}}{Add} \overset{\Gamma\vdash A \qquad \Gamma\vdash B}{\text{WeChat powcoder}} \land \text{I}$

$$\frac{\Gamma \vdash A \land B}{\Gamma \vdash A} \land -\text{E}_1 \qquad \frac{\Gamma \vdash A \land B}{\Gamma \vdash B} \land -\text{E}_2$$

More rules

Disjunction (or) has two introduction rules:

Assignment Project Exam Help

Disjunction elimination is a little unusual:

The true literal, written \top , has only an introduction:

Add WeChat powcoder

And false, written \perp , has just elimination (ex falso quodlibet):

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash P}$$

Example

ProvAssignment Project Exam Help

https://powcoder.com

Example

ProvAssignment Project Exam Help

- \bullet $A \lor \bot \to A$

https://powcoder.com

Example

ProvAssignment Project Exam Help

- \bullet $A \lor \bot \to A$
- What would ne atto p Squ val pro Wcoder.com

Example

ProvAssignment Project Exam Help

- \bullet $A \lor \bot \to A$

What would ne htto psqu/val prowcoder.com Typically we just define

$$\neg A \equiv (A \rightarrow \bot)$$

Add WeChat powcoder

Example

Prove:

 \bullet $A \rightarrow (\neg \neg A)$

Algebraic Type Isomorphism

Example

ProvAssignment Project Exam Help

- \bullet $A \lor \bot \to A$

What would ne htto psqu/val prowcoder.com Typically we just define

$$\neg A \equiv (A \rightarrow \bot)$$

Add WeChat powcoder

Example

Prove:

- \bullet $A \rightarrow (\neg \neg A)$
- \bullet $(\neg \neg A) \rightarrow A$

Example

ProvAssignment Project Exam Help

- \bullet $A \lor \bot \to A$

What would ne htto psqu/val prowcoder.com Typically we just define

$$eg A \equiv (A
ightarrow oldsymbol{\perp})$$

Add WeChat powcoder

Example

Prove:

- \bullet $A \rightarrow (\neg \neg A)$
- $(\neg \neg A) \rightarrow A$ We get stuck here!

Constructive Logic

The Aissigenment resolution the Examental design.

 $P \vee \neg P$

https://powcoder.com

Add WeChat powcoder

This is because it is a *constructive* logic that does not allow us to do proof by contradiction.

Boiling Haskell Down

The theoretical properties we will describe also apply to Haskell, but we reed a smaller language so described by the purposes. To ject Exam Help

- No user-defined types, just a small set of built-in types.
- No polymorphism (type variables)
- Just lambattps: defiponwcoderiacom

This language is a very minimal functional language, called the simply typed lambda calculus, originally due to Alonzo Church.

Our small set of built-in types are intended to be enough to express most of the data types we would otherwise define.

We are going to use logical inference rules to specify how expressions are given types (*typing rules*).

Function Types

we Assignment, Project Exams Help

 $x :: A, \Gamma \vdash e :: B$ https://powcoder.com

The typing rule for function application is as follows:

Add WeChat powcoder

What other types would be needed?

In addis Sife the control of the con

https://powcoder.com

In addis Sife the control of the con

https://powcoder.com

Add WeChat powcoder

Records

Algebraic Type Isomorphism

In addis Soft and the soft of the soft of

https://powcoder.com

Add WeChat powcoder

Records

In addis Sife the control of the con

https://powcoder.com

Add WeChat powcoder

Records

Combining values conjunctively

We want to store two things in one value.

(might wAt Sessignification this one) Project Exam Help

Haskell Inples S:// powcoder.com

type Point = (Float, Float)

midpoint (x1, yAccolor)

= ((x1+x2)/2, (y1+y2)/2)

midpoint (x1, yAccolor)

midpoint (x1, yAccolor)

= ((x1+x2)/2, (y1+y2)/2)

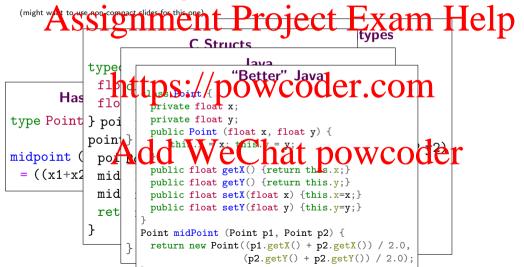
Combining values conjunctively



Combining values conjunctively

```
Assignment Project Exam Help
                                                                                                                                                                                     C Structs
                                                                               typedef struct point {
                                                                                      flattps://powcoder.com
  type Point } point;
midpoint (point midPoint (point p1) point p2) {
point all WeChat power of the point p2) {
point all power of the point p3) {
point all power of the passes o
            = ((x1+x2) \text{ mid.x} = (p1.x + p2.x) / 2.0;
                                                                                        mid.v = (p2.v + p2.v) / 2.0;
                                                                                         return mid:
```

```
(might wAt to use appropriate lides for this one) the Project Exam Help
                      C Structs
                            Java
             Htps://powcoder.com
type Point } poi
              public float y;
         poin \}
midpoint por addid the Coinat powcoder
 = ((x1+x2)
          mid
               Point mid = new Point():
          mid
               mid.x = (p1.x + p2.x) / 2.0;
               mid.y = (p2.y + p2.y) / 2.0;
               return mid:
```



Product Types

For simply typed lambda calculus, we will accomplish this with tuples, also called product Exam Help

https://powceder.com

We won't have the declarations named field or anything like that. More than two values can be combined by nesting products, for example a three dimensional vector:

(Int,(Int,Int))

Constructors and Eliminators

we Assignment Project Exam Help

https://poweoder.com

The only way to extract each component of the product is to use the fst and snd eliminators:

Unit Types

Assignment Project Exam Help

Currently, we have no way to express a type with just one value. This may seem useless at first, but it becomes useful in combination with other types. We'll introduce the trip spe/fyop the with other types.

We'll introduce the trip spe/fyop the with other types.

Add WeChatopowcoder

Recap: Logic

Disjunctive Composition

We can't, with the types we have, express a type with exactly three values.

data Traffic Light = Red | Amber | Green | Exam Help

https://powcoder.com

Disjunctive Composition

We can't, with the types we have, express a type with exactly three values.

data Traffic Light = Red | Amber | Green | Exam Help

In general we want to express data that can be one of multiple alternatives, that contain different ptt $\frac{1}{2}$ $\frac{1}{2}$

Example (More elaborate alternatives)

```
type Length = Int
type Angle = Atd We hat powcoder
data Shape = Rect Length Length | Point
```

Triangle Angle Length Length

This is awkward in many languages. In Java we'd have to use inheritance. In C we'd have to use unions.

Sum Types

We'll build in the Haskell Either type to express the possibility that data may be one of twelvestignment $Project\ Exam\ Help$

https://powegoder.com

These types are Also alled we that powcoder

Our TrafficLight type can be expressed (grotesquely) as a sum of units:

 $TrafficLight \simeq Either () (Either () ())$

Constructors and Eliminators for Sums





We can branch based on which alternative is used using pattern matching:



Examples

Assignment Project Exam Help

Our traffic light type has three values as required:

https://poweoder.eom



Assignment Project Exam Help

We add another type, called Void, that has no inhabitants. Because it is empty, there

is no way to construct it.
We do have a wate project in the construct it.

 $\Gamma \vdash e :: Void$

The Empty Type

We add another than the land of the land o

We do have a way to eliminate it, however:

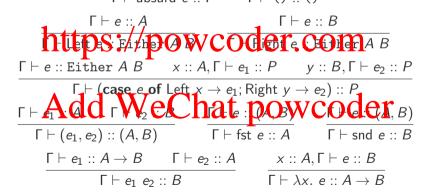
https://powcoder.com

 $\Gamma \vdash \text{absurd } e :: P$

If I have a variable of the environment of the envi

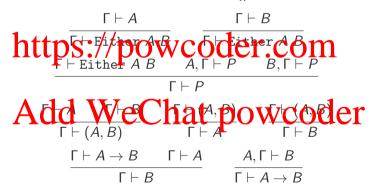
Gathering Rules

Assignmente Project Exam Help



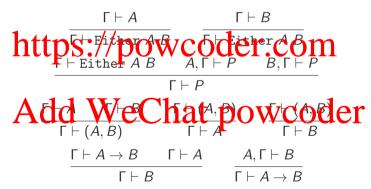
Removing Terms...

Assignment Project Exam Help



Removing Terms...

Assignment Project Exam Help



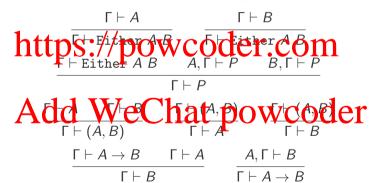
This looks exactly like constructive logic!

37

Removing Terms...

Algebraic Type Isomorphism

Assignment Project Exam Help



This looks exactly like constructive logic!

If we can construct a program of a certain type, we have also created a proof of a

This correspondence goes by many names, but is usually attributed to Haskell Curry and William Floward ment Project Exam Help

	Programming	Logic	
1 44	Туреs	Propositions	
http	S Programs V	vcoder.co	m
P	Evaluation	VCOP off .CO	

The Curry-Howard Correspondence

This correspondence goes by many names, but is usually attributed to Haskell Curry and William Howard ment Project Exam Help

	Programming	Logic	
1 44	Туреs	Propositions	
httr	S Programs V	vcoder.co	\mathbf{m}
	Evaluation	VCOP off. CO Proof Simplification	

It turns out, no matter what logic you want to define, there is always a corresponding λ -calculus, and Λ ce ders.

Typed λ-Calculus
Continuations
Continuations
Monads
Linear Types, Session Types
Region Types

Constructive Logic
Classical Logic
Modal Logic
Linear Logic
Separation Logic

Example Seignment Project Exam Help

and Comm :: $(A, B) \rightarrow (B, A)$ and Comm p = (snd p, fst p)This proves A **https:**//powcoder.com

Exam Help

and Comm :: $(A, B) \rightarrow (B, A)$

This proves A https://powcoder.com

Example (Transitivity of Implication) Add WeChat powcoder

Exam Help

and Comm :: $(A, B) \rightarrow (B, A)$

This proves A https://powcoder.com

Example (Transitivity of Implication) Add::WeChat powcoder

Exam Help

and Comm :: $(A, B) \rightarrow (B, A)$ and Comm p = (snd p, fst p)This proves A by A powcoder.com

Example (Transitivity of Implication)

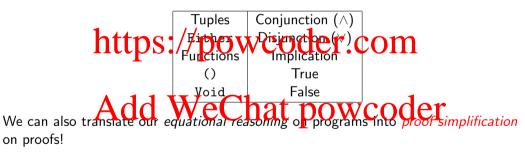
AdditiWeChat powcoder

transitive f g x = g (f x)

Transitivity of implication is just function composition.

Translating

Assignment Project Exam Help



Assument Project Exam Help We have this unpleasant proof:

> https://poweoder.com Add WeChat bowcoder

Transaing Signment Project Exam Help Assuming x :: (A, B), we want to construct (B, A).

https://pow.coder.com

Add WeChat powcoder

Transaing Signment Project Exam Help Assuming x :: (A, B), we want to construct (B, A).

Transaissignment Project Exam Help Assuming x :: (A, B), we want to construct (B, A).

https://powcoder.com $\mathbf{x} :: (A, B)$ Add WeChat powcoder

Transating Signment Project Exam Help Assuming x :: (A, B), we want to construct (B, A).

https://powcoder $\frac{x :: (A, B)}{Add} = \frac{(fst \times fst \times) :: (A, A)}{(fst \times fst \times) :: (A, A)}$ Add We Chat powcoder

Transaissignment Project Exam Help Assuming x :: (A, B), we want to construct (B, A).

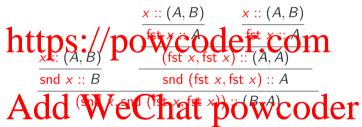
https://powcoder.com x :: (A, B) (fst x, fst x) :: (A, A)Add WeChat powcoder

Recap: Logic

Transating Signament Project Exam Help Assuming x :: (A, B), we want to construct (B, A).

 $\begin{array}{c} https://po_{fst}^{x}, (A,B) \\ \xrightarrow{x:: (A,B)} & (fst \ x, fst \ x) :: (A,A) \\ Add (snd x, snd (fst \ x, fst \ x) :: (A,B) \\ & (fst \ x, fst \ x) :: (A,A) \\ \end{array}$

Assum SSI gn meent enterprise Exam Help



We know that

$$(\operatorname{snd} x, \operatorname{snd} (\operatorname{fst} x, \operatorname{fst} x)) = (\operatorname{snd} x, \operatorname{fst} x)$$

Lets apply this simplification to our proof!

AssuArssignment Project Exam Help

https://poxxecoefer.com

AssuAnssignment Project Exam Help

https://poxxcoder.com

Back to logic:

Add Wethat bowcoder

Applications

As ments albert, in content to the distinction between value-level and type-level languages is removed, allowing us to refer to our program in types (i.e. propositions) and then construct programs of those types (i.e. proofs) types (i.e. proofs)

Peano Arithmetic

If there's time, Liam will demo how to prove some basic facts of natural numbers in Agda, a dependently pred applies that powcoder

Generally, dependent types allow us to use rich types not just for programming, but also for verification via the Curry-Howard correspondence.

Caveats

All functions we define have to be total and terminating.

Otherwise weiget an inconsistent of that lets us project also things: Help

 $proof_1 : P = NP$ $proof_1 = proof_1$

https://powcoder.com

 $proof_2 = proof_2$

Most common calculi correspond to constructive pogic, not classical ones, so principles like the law of excluded middle or double negation elimination do not hold:

These types we have defined form an algebraic structure called a *commutative*

SemirAs Signment Project Exam Help

• Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)

https://powcoder.com

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $\overset{A}{h} \cong \overset{A}{powcoder.com}$

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either AB) $C \simeq$ Either A (Either BC)
- Identity: Either Void A ~ A
 Commutate to the control of the con

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $A \simeq A$ Commutate The Ser A DOWNER COM

Laws for tuples and 1

• Associativity: $((A, B), C) \simeq (A, (B, C))$

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $A \simeq A$ Commutate The Ser A DOWNER COM

Laws for tuples and 1

- Associativity: $((A, B), C) \simeq (A, (B, C))$
- Identity: (Add WeChat powcoder

These types we have defined form an algebraic structure called a *commutative*

SemirAge Signment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $A \simeq A$ Commutate The Ser A DOWNER COM
- Laws for tuples and 1
 - Associativity: $((A, B), C) \simeq (A, (B, C))$
 - Identity: (AA) AWe Chat powcoder
 Commutativity: (A,B) \(\subseteq (B,A) \)

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $A \simeq A$ Commutate The Ser A DOWNER COM
- Laws for tuples and 1
 - Associativity: $((A, B), C) \simeq (A, (B, C))$
 - Identity: (AA) AWe Chat powcoder
 Commutativity: (A,B) \(\sum_{(B,A)} \) that powcoder

Combining the two:

• Distributivity: $(A, \text{Either } B \ C) \simeq \text{Either } (A, B) \ (A, C)$

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $A \simeq A$ Commutate The Ser A DOWNER COM
- Laws for tuples and 1
 - Associativity: $((A, B), C) \simeq (A, (B, C))$
 - Identity: (AA) AWe Chat powcoder
 Commutativity: (A,B) \(\sum_{(B,A)} \) that powcoder

Combining the two:

- Distributivity: $(A, \text{Either } B \ C) \simeq \text{Either } (A, B) \ (A, C)$
- Absorption: (Void, A) \simeq Void

These types we have defined form an algebraic structure called a *commutative*

SemirAge Ssignment Project Exam Help

- Associativity: Either (Either A B) $C \simeq$ Either A (Either B C)
- Identity: Either Void $A \simeq A$ Commutate The Ser A DOWNER COM

Laws for tuples and 1

- Associativity: $((A, B), C) \simeq (A, (B, C))$
- Identity: (AA) AWe Chat powcoder
 Commutativity: (A,B) \(\sum_{(B,A)} \) that powcoder

Combining the two:

- Distributivity: $(A, \text{Either } B \ C) \simeq \text{Either } (A, B) \ (A, C)$
- Absorption: (Void, A) \simeq Void

What does \simeq mean here? It's more than logical equivalence.

Isomorphism

Two types A and B are isomorphic, written $A \simeq B$, if there exists a bijection between them. This means that for each value in A we can find unique value in B and vice versa.

Example (Refactoring)

We can use this reasoning to simplify type definitions. For example data Switch Int POWCOGET. COM

Off Name

Can be simplified to the isomorphic (Name, Maybe Int).

Add We Chat powcoder

66

Recap: Logic

Isomorphism

Two types A and B are isomorphic, written $A \simeq B$, if there exists a bijection between them This means that for each value in A we can find a unique value in B and vice versa.

Example (Refactoring)

We can use this reasoning to simplify type definitions. For example data Switch Int POWCOGET. COM

Off Name

Can be simplified to the isomorphic (Name, Maybe Int).

Add We Chat powcoder

Generic Programming

Representing data types generically as sums and products is the foundation for generic programming libraries such as GHC generics. This allows us to define algorithms that work on arbitrary data structures.

Consider the type of fst: fst A Sa ign ment Project Exam Help

This can be written more verbosely as:

Or, in a more matternation of the order of t

fst ::
$$\forall a \ b. \ (a,b) \rightarrow a$$

This kind of quantification over the Calabastis play was polymorphism for short.

(It's also called generics in some languages, but this terminology is bad)

What is the analogue of \forall in logic? (via Curry-Howard)?

Assignment Project Exam Help

The type quantifier to gesponds to a wive a quantifier of the same as the \forall from first-order logic. What's the difference?

Add WeChat powcoder

Recap: Logic

Curry-Howard

The type quantifier \forall corresponds to a universal quantifier \forall , but it is not the same as the **Act Site of the Control of**

First-order logic quantifiers range over a set of *individuals* or values, for example the natural numbers:

https://powcoder.com

These quantifiers range over propositions (types) themselves. It is analogous to second-order logic, not first-order:

 $Add \ \underset{\forall A. \ \forall B. \ (A,B) \ \rightarrow \ (B,A)}{ \ \ \forall A. \ \forall B. \ (A,B) \ \rightarrow \ (B,A)}$

The first-order quantifier has a type-theoretic analogue too (type indices), but this is not nearly as common as polymorphism.

Generality

If we feed a function of type Int — Int. a polymorphic function of type $\forall a.1a \rightarrow a$ will do just line we can just instantiate the type variable to first. But the reverse is not true. This gives rise to an ordering.

https://powcoder.com

If we need a function of type Int Dipt, a polymorphic function of type $\forall a.1a \rightarrow a$ will do just fine we can just instantiate the type variable to firt. But the lever e is not

true. This gives rise to an ordering.

Generality
A type A is more general transformation and appears of the instantiated to give the type B.

Add WeChat powcoder

Recap: Logic

Generality

If we need a function of type Int

— polymerphic function of type Value a polymerphic function of type Value a will do just line we can just instantiate the type variable to Int. But the reverse is not true. This gives rise to an ordering.

Generality
A type A is more general transformation and appears of the instantiated to give the type B.

Example (Fundamental WeChat powcoder

Int \rightarrow Int \supseteq $\forall z. z \rightarrow z$ \supseteq $\forall x y. x \rightarrow y$ \supseteq $\forall a. a$

Assignment Project Exam Help

How many possible total, terminating implementations are there of a function of the following type?

https://powcoder.com

Typed Lambda Calculus

Constraining Implementations

Assignment Project Exam Help

How many possible total, terminating implementations are there of a function of the following type?

https://powcoder.com

Constraining Implementations

Assignment Project Exam Help

How many possible total, terminating implementations are there of a function of the following type?

https://powcoder.com

Add We Chat powcoder

Polymorphic type signatures constrain implementations.

Parametricity

The principle of parametricity states that the result of polymorphic functions cannot

The principle of parametricity states that the result of polymorphic functions cannot depend on values of an abstracted type.

Example

Add WeChat powcoder

foo :: $\forall a. [a] \rightarrow [a]$

Parametricity

Definition

The Angrephinien the President of polynamoful cities and the depend on values of an abstracted type.

More formally, suppose I have a polymorphic function g that is polymorphic on type a. If run any arbitrary function $f: a \to a$ on all the a values in the input of g, that will give the same result is soming printing that the County of the output.

Example

Add WeChat-powcoder

We know that every element of the output occurs in the input. The parametricity theorem we get is, for all f:

 $foo \circ (map \ f) = (map \ f) \circ foo$

More Examples

Assignment Project Exam Help

https://pewcoder.com

What's the parametricity theorems? Add WeChat powcoder

More Examples

Assignment Project Exam Help

What's the paractipes the proposed of the proposed of the paractic prop

Example (Answer)

For any f:

Typed Lambda Calculus

More Examples

Assignment Project Exam Help

https://powcoder.com

What's the parametricity theorem? Add WeChat powcoder

Assignment Project Exam Help

 $(++):: \forall a. \ [a] \rightarrow [a] \rightarrow [a]$

What's the parametric shove powcoder.com

Example (Answer)

Add-WeChat powcoder

Recap: Logic

Typed Lambda Calculus

More Examples

Assignment Project Exam Help

https://powcoder.com

What's the parametricity theorem? Add WeChat powcoder

More Examples

Assignment Project Exam Help

concat :: $\forall a$. $[[a]] \rightarrow [a]$

What's the parametric shove powcoder.com

Example (Answer)

Higher Order Functions

Assignment Project Exam Help

https://paweoder.com

What's the parametricity theorem?

Typed Lambda Calculus

Higher Order Functions

Assignment Project Exam Help

 $\textit{filter} :: \forall \textit{a}. \ (\textit{a} \rightarrow \textit{Bool}) \ \rightarrow [\textit{a}] \rightarrow [\textit{a}]$

What's the parinttips represented by the property of the parinttips of the property of the parint of

Example (Answer)

Parametricity Theorems

Assignment Project Exam Help

Follow a similar structure. In fact it can be mechanically derived, using the relational parametricity francion Sventer OW C. Roman and Oppraised by Wadler in the famous paper, "Theorems for Free!" I.

Upshot: We can ask lambdabot on the Haskell IRC channel for these theorems.

¹https://people.mpi-sws.org/~dreyer/tor/papers/wadler.pdf

Wrap-up

Assignment Project Exam Help

- That's the entirety of the assessable course content for COMP3141.
- There is a quiz for this week, but no exercise (there's still Assignment 2)

 Next week's lectures consist of a extension recture video on dependent type
- systems, and a revision lecture on Wednesday with Curtis...
- Please come up with questions to ask Curtis for the revision lecture! It will be over very quite there exists ask Curtis for the revision lecture! It will be