Assignment Project Exam Help

COMPSCI 141

https://eduassistpro.github.io/

Liam O'Conn
University of Edinburgh, LFCS (an

Add WeChat edu_assist_pro

Term 2, 2020

Motivation

Assignment Project Exam Help

We've already see

https://eduassistpro.github.io/

How do we come up with correctness properties in the first place?

Add WeChat edu_assist_pro

Structure of a Module

A Haskell program will usually be made up of many modules, each of which exports one or more *data types*.
Typically a module for a data type `X` will also provide a set of functions, called *operations*, on

- `X`
- `X`
- to update the data type:

A lot of software can be designed with this structure.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Example (Data Types)

A dictionary data type, with empty, insert and lookup.

Data Invariants

Assignment Project Exam Help

One source of properties is *data invariants*.

Data Invariants

Data invariants a

Whenever we use a

invariants are mai

https://eduassistpro.github.io/

Example

Add WeChat edu_assist_pro

- That a list of words in a dictionary is always in sorted order.
- That a binary tree satisfies the search tree properties.
- That a date value will never be invalid (e.g. 31/13/2019).

Properties for Data Invariants

For a given data type `X`, we define a *wellformedness predicate*

$$\texttt{wf} :: \texttt{X} \to \texttt{Bool}$$

For a given value x.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Properties

For each operation, if all input values of type `X` satisfy `wf`.

In other words, for each constructor operation c $\texttt{wf}\,(c\ \cdots)$, and for each update operation $u :: \texttt{X} \to \texttt{X}$ we must show $\texttt{wf}\ x \implies \texttt{wf}(u\ x)$

Demo: Dictionary example, sorted order.

Stopping External Tampering

Assignment Project Exam Help

Even with our sorted dictionary example, there's nothing to stop a malicious or clueless programmer fro

https://eduassistpro.github.io/

Example

The malicious programmer could just add a word directly to the dictionary, unsorted, bypassing our carefully written `insert` function.

Add WeChat edu_assist_pro

We want to prevent this sort of thing from happening.

Abstract Data Types

An *abstract* data type (ADT) is a data type where the implementation details of the type and its associated operations are hidden.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Abstract Data Types

An *abstract* data type (ADT) is a data type where the implementation details of the type and its associated operations are hidden.

Assignment Project Exam Help

```
newtype Dict
type Word = Strin
type Definit
emptyDict   :: Dict
insertWord  :: Word -> Definition -> Dict -> Dict
lookup      :: Word -> Dict -> Maybe Definition
```

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

If we don't have access to the implementation of the provided operations, which we know preserve our data inv invariants cannot be violated if this module is correct.

Abstract Data Types

Assignment Project Exam Help

An *abstract* data type (ADT) is a data type where the implementation details of the type and its associated operations are hidden.

https://eduassistpro.github.io/

```
newtype Dict
type Word = Strin
type Definit
emptyDict   :: Dict
insertWord :: Word -> Definition -> Dict -> Dict
lookup      :: Word -> Dict -> Maybe Definition
```

Add WeChat edu_assist_pro

If we don't have access to the implementation of the provided operations, which we know preserve our data inv invariants cannot be violated if this module is correct.

Demo: In Haskell, we make ADTs with module headers.

Abstract? Data Types

Assignment Project Exam Help

In general, *abs*

The inverse of abs

https://eduassistpro.github.io/

Abstract data typ

implementation details are hidden, and we no longer have to re

level of implementation.

Add WeChat edu_assist_pro

Validation

Assignment Project Exam Help

Suppose we had a `sendEmail` function:

```
sendEmail :: String -- email address
          -
          -
```

https://eduassistpro.github.io/

It is possible to mix th t, it's possible that the given email address is not valid.

Question

Add WeChat edu_assist_pro

Suppose that we wanted to make it impossible to call t first checking that the email address was valid.
How would we accomplish this?

Validation ADTs

We could define a tiny ADT for validated email addresses, where the data invariant is that the contained email address is valid:

Assignment Project Exam Help

```
module EmailADT(Email, checkEmail, sendEmail)
    newtype E

    checkEm
    checkEm
                | otherwise      = Nothing
```

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Validation ADTs

We could define a tiny ADT for validated email addresses, where the data invariant is that the contained email address is valid:

Assignment Project Exam Help

```
module EmailADT(Email, checkEmail, sendEmail)
    newtype E

    checkEm
    checkEm
                  | otherwise      = Nothing
```

https://eduassistpro.github.io/

Then, change the type of sendEmail:

Add WeChat edu_assist_pro

```
    sendEmail :: Email -> String -> IO()
```

Validation ADTs

We could define a tiny ADT for validated email addresses, where the data invariant is that the contained email address is valid:

Assignment Project Exam Help

```
module EmailADT(Email, checkEmail, sendEmail)
    newtype E

    checkEm
    checkEm
                  | otherwise       = Nothing
```

https://eduassistpro.github.io/

Then, change the type of `sendEmail`:

Add WeChat edu_assist_pro

```
sendEmail :: Email -> String -> IO()
```

The only way (outside of the `EmailADT` module) to create a value of type `Email` is to use `checkEmail`.

Validation ADTs

We could define a tiny ADT for validated email addresses, where the data invariant is that the contained email address is valid:

Assignment Project Exam Help

```
module EmailADT(Email, checkEmail, sendEmail)
    newtype E

    checkEm
    checkEm
                  | otherwise       = Nothing
```

https://eduassistpro.github.io/

Then, change the type of `sendEmail`:

Add WeChat edu_assist_pro

```
sendEmail :: Email -> String -> IO()
```

The only way (outside of the `EmailADT` module) to create a value of type `Email` is to use `checkEmail`.

`checkEmail` is an example of what we call a *smart constructor*: a constructor that enforces data invariants.

Reasoning about ADTs

Consider the following, more traditional example of an ADT interface, the unbounded queue:

Assignment Project Exam Help

```
data Queue

emptyQueue :: Q
enqueue :: Int -> Q
front   :: Queue -> Int
dequeue :: Queue -> Queue -- partial
size    :: Queue -> Int
```

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

We could try to come up with properties that relate these functio without reference to their implementation, such as:

```
dequeue (enqueue x emptyQueue) == emptyQueue
```

However these do not capture functional correctness (usually).

Models for ADTs

We could imagine a simple implementation for queues, just in terms of lists:

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Models for ADTs

We could imagine a simple implementation for queues, just in terms of lists:

```
emptyQueueL = []
enqueueL a  = (++ [a])
frontL      = head
dequeueL    = tail
sizeL       = length
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Models for ADTs

We could imagine a simple implementation for queues, just in terms of lists:

```
emptyQueueL = []
enqueueL a  = (++ [a])
frontL      = head
dequeueL    = tail
sizeL       = length
```

But this implementation is $\mathcal{O}(n)$ to enqueue! Unacce

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Models for ADTs

We could imagine a simple implementation for queues, just in terms of lists:

```
emptyQueueL = []
enqueueL a  = (++ [a])
frontL      = head
dequeueL    = tail
sizeL       = length
```

But this implementation is $\mathcal{O}(n)$ to enqueue! Unacce

However!

This is a dead simple implementation, and trivial to see that it is cor
better queue implementation, it should always give the same results as this simple one. Therefore: This implementation serves as a functional correctness specification for our `Queue` type!

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Refinement Relations

The typical approach to connect our model queue to our `Queue` type is to define a relation, called a *refinement relation*, that relates a `Queue` to a list and tells us if the two structures represent the same queue conceptually:

```
rel :: Queue -> [In
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Refinement Relations

The typical approach to connect our model queue to our `Queue` type is to define a relation, called a *refinement relation*, that relates a `Queue` to a list and tells us if the two structures represent the same queue conceptually:

```
rel :: Queue -> [In
```

Then, we show tha

```
prop_empty_
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Refinement Relations

The typical approach to connect our model queue to our `Queue` type is to define a relation, called a *refinement relation*, that relates a `Queue` to a list and tells us if the two structures represent the same queue conceptually:

```
rel :: Queue -> [In
```

Then, we show tha

```
prop_empty_
```

That any query functions for our two types produce equal result
such as for size:

```
prop_size_r fq lq = rel fq lq ==> size fq == size lq
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Refinement Relations

The typical approach to connect our model queue to our `Queue` type is to define a relation, called a *refinement relation*, that relates a `Queue` to a list and tells us if the two structures represent the same queue conceptually:

```
rel :: Queue -> [In
```

Then, we show tha

```
prop_empty_
```

That any query functions for our two types produce equal result
such as for size:

```
prop_size_r fq lq = rel fq lq ==> size fq = size lq
```

And that each of the queue operations preserves our refinement relation, for example for enqueue:

```
prop_enq_ref fq lq x =
  rel fq lq ==> rel (enqueue x fq) (enqueueL x lq)
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

In Pictures

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

In Pictures

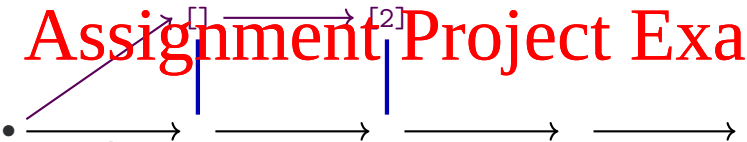
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
prop_empty_r = rel emptyQueue emptyQueue
```

In Pictures

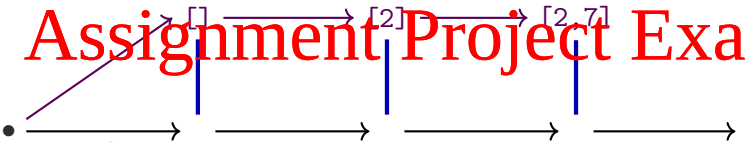


Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
prop_enq ref fq lq x =
rel fq lq ==> ref (enqueue x fq) (enqueue x lq)
```

In Pictures

Assignment Project Exam Help

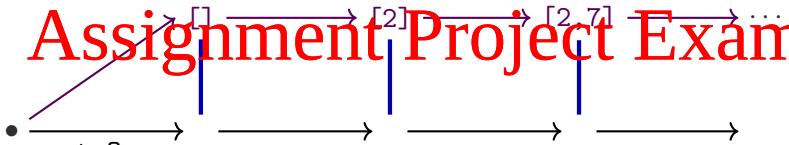


https://eduassistpro.github.io/

```
                    prop_enq ref fq lq x =
rel fq lq ==> ref (enqueue x fq) (enqueue x lq)
```

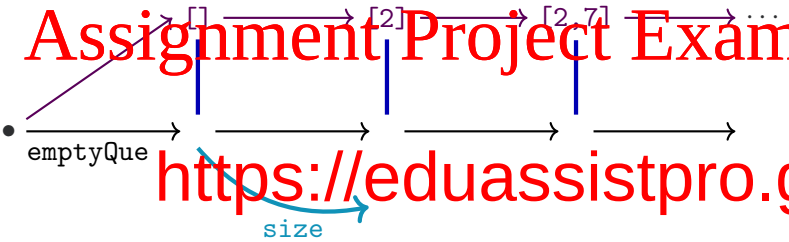
Add WeChat edu_assist_pro

In Pictures



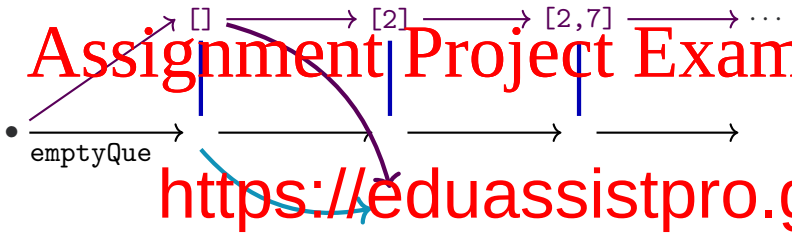
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

In Pictures



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

In Pictures



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
prop_size_r fq lq = rel fq lq ==> size fq == sizeL lq
```

In Pictures

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
prop_size_r fq lq = rel fq lq ==> size fq == sizeL lq
```

Whenever we use a Queue, we can reason as if it were a list!

Abstraction Functions

Assignment Project Exam Help

These refinement relations are very difficult to use with QuickCheck because the `rel fq lq` preconditions are very hard to satisfy with randomly generated inputs.

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Abstraction Functions

Assignment Project Exam Help

These refinement relations are very difficult to use with QuickCheck because the `rel fq lq` preconditions are very hard to satisfy with randomly generated inputs.

For this example, it
the correspondin

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Abstraction Functions

Assignment Project Exam Help

These refinement relations are very difficult to use with QuickCheck because the `rel fq lq` preconditions are very hard to satisfy with randomly generated inputs.

For this example, it

the correspondin

https://eduassistpro.github.io/

Conceptually, our refinement relation is then just:

Add WeChat edu_assist_pro

```
\fq lq → absfun fq =
```

However, we can re-express our properties in a much more QC-friendly format (**Demo**)

Fast Queues

Assignment Project Exam Help

Let's use test-driven development! We'll implement a fast Queue with amortised $\mathcal{O}(1)$ operations.

https://eduassistpro.github.io/

```
data Queue
       Int    -- size of the rear
```

We store the rear part of the queue in reverse order, to make enque

Add WeChat edu_assist_pro

Fast Queues

Assignment Project Exam Help

Let's use test-driven development! We'll implement a fast Queue with amortised $\mathcal{O}(1)$ operations.

```
data Queue
```

https://eduassistpro.github.io/

```
       Int    -- size of the rear
```

We store the rear part of the queue in reverse order, to make enque

Add WeChat edu_assist_pro

Thus, converting from our Queue to an abstract list re

```
toAbstract :: Queue -> [Int]
toAbstract (Q f sf r sr) = f ++ reverse r
```

Data Refinement

These kinds of properties establish what is known as a *data refinement* from the abstract, slow list model to the fast, complex Queue implementation.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Data Refinement

These kinds of properties establish what is known as a *data refinement* from the abstract, slow list model to the fast, concrete Queue implementation.

Assignment Project Exam Help

Refinement and Specifications

In general, all funct

1. all data inva

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Data Refinement

These kinds of properties establish what is known as a *data refinement* from the abstract, slow list model to the fast, concrete Queue implementation.

Assignment Project Exam Help

Refinement and Specifications

In general, all funct

1. all data inva
2. the implem

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Data Refinement

These kinds of properties establish what is known as a *data refinement* from the abstract, slow list model to the fast, complex queue implementation.

Assignment Project Exam Help

Refinement and Specifications

In general, all funct

1. all data inv
2. the implem

https://eduassistpro.github.io/

There is a limit to the amount of abstraction we can do before they b testing (but not necessarily for proving).

Add WeChat edu_assist_pro

Data Refinement

These kinds of properties establish what is known as a *data refinement* from the abstract, slow list model to the fast, complex queue implementation.

Assignment Project Exam Help

Refinement and Specifications

In general, all funct

1. all data inv
2. the implem

https://eduassistpro.github.io/

There is a limit to the amount of abstraction we can do before they b testing (but not necessarily for proving).

Add WeChat edu_assist_pro

Warning

While abstraction can simplify proofs, abstraction does not reduce the fundamental complexity of verification, which is provably hard.

Data Invariants for Queue

In addition to the already-stated refinement properties, we also have some data invariants to maintain for a value `Q f sf rf ef`:

1. `length f == sf`

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Data Invariants for Queue

In addition to the already-stated refinement properties, we also have some data invariants to maintain for a value `Q f sf r sr`:

1. `length f == sf`
2. `length r == sr`

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Data Invariants for Queue

In addition to the already-stated refinement properties, we also have some data invariants to maintain for a value `Q f sf r sr`:

1. `length f == sf`
2. `length r == sr`
3. **importan**

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Data Invariants for Queue

In addition to the already-stated refinement properties, we also have some data invariants to maintain for a value `Q f sf r sr`:

1. `length f == sf`
2. `length r == sr`
3. **importan**

We will ensure our ese invariants.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Data Invariants for Queue

In addition to the already-stated refinement properties, we also have some data invariants to maintain for a value `Q f sf r sr`:

1. `length f == sf`
2. `length r == sr`
3. **importan**

We will ensure our ese invariants.

Thus, our `wellformed` predicate is used merely to enfo outputs of our operations:

```
prop_wf_empty = wellformed (emptyQueue)
prop_wf_enq q = wellformed (enqueue x q)
prop_wf_deq q = size q > 0 ==> wellformed (dequeue q)
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Implementing the Queue

Assignment Project Exam Help

We will generally implement by:

- Dequeue fr
- Enqueue to t
- If necessary
  and appending it to the front.

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Implementing the Queue

Assignment Project Exam Help

We will generally implement by:

- Dequeue fr
- Enqueue to t
- If necessary
  and appending it to the front.

  This step is slow ($\mathcal{O}(n)$), but only happens every g
  average case amortised complexity of $\mathcal{O}(1)$ ti

https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of [1..7] to the emptyQueue in turn:

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Q	[]	0	[]	0

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of [1..7] to the emptyQueue in turn:

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of [1..7] to the emptyQueue in turn:

```
Q []          0 []      0
                        0 (
                        1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of [1..7] to the emptyQueue in turn:

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of [1..7] to the emptyQueue in turn:

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Q		0		0
				0 (
				1

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of [1..7] to the emptyQueue in turn:

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

	Q []	0	[]	0
				0 (
				1
→	Q [1, 2, 3]	3	[5	

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of $[1..7]$ to the emptyQueue in turn:

	Q	[]	0	[]
$\rightarrow$	Q	$[1,2,3]$	3	[5
$\rightarrow$	Q	$[1,2,3]$	3	[6

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of $[1..7]$ to the emptyQueue in turn:

Assignment Project Exam Help
https://eduassistpro.github.io/
AddWeChat edu_assist_pro

	Q	[]	0	[]	0
					0 (
					1
$\rightarrow$	Q	$[1,2,3]$	3	[5	
$\rightarrow$	Q	$[1,2,3]$	3	[6	
$\rightarrow$	Q	$[1,2,3,4,5,6,7]$	7	[]	0 (

Amortised Cost

```
enqueue x (Q f sf r sr) = inv3 (Q f sf (x:r) (sr + 1))
```

When we enqueue each of $[1..7]$ to the empty queue in turn:

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

	Q []	0	[]	0
				0 (
				1
$\rightarrow$	Q [1, 2, 3]	3	[5	
$\rightarrow$	Q [1, 2, 3]	3	[6	
$\rightarrow$	Q [1, 2, 3, 4, 5, 6, 7]	7	[]	0 (

Observe that the slow invariant-reestablishing step $(*)$ happens after 1 step, then 2, then 4...

Extended out, this averages out to $\mathcal{O}(1)$.

Another Example

Consider this ADT interface for a bag of numbers:

```
data Bag
emptyBag   :: Bag
addToBag   :: Int -> Ba
averageBag :: B
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Another Example

Consider this ADT interface for a bag of numbers:

```
data Bag
emptyBag    :: Bag
addToBag    :: Int -> Ba
averageBag  :: B
```

Our conceptual a

```
emptyBagA = []

addToBagA x xs = x:xs

averageBagA [] = Nothing
averageBagA xs = Just (sum xs `div` length xs)
```

But do we need to keep track of all that information in our implementation?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Another Example

Consider this ADT interface for a bag of numbers:

```
data Bag
emptyBag    :: Bag
addToBag    :: Int -> Ba
averageBag  :: B
```

Assignment Project Exam Help
https://eduassistpro.github.io/

Our conceptual a

```
emptyBagA = []

addToBagA x xs = x:xs

averageBagA [] = Nothing
averageBagA xs = Just (sum xs `div` length xs)
```

Add WeChat edu_assist_pro

But do we need to keep track of all that information in our implementation? No!

Concrete Implementation

Assignment Project Exam Help

Our concrete version will just maintain two integers, the total and the count:

```
data Bag = B { total :: Int , count :: Int }
emptyBag :: Bag
emptyBag = B 0 0

addToBag :: Int -> Bag -> Bag
addToBag x (B t c) = B (x + t) (c + 1)

averageBag :: Bag -> Maybe Int
averageBag (B _ 0) = Nothing
averageBag (B t c) = Just (t `div` c)
```

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Refinement Functions

Assignment Project Exam Help

Normally, writing an abstraction function (as we did for `Queue`) is a good way to express our refine

write such a functi

https://eduassistpro.github.io/

```
toAbstract :: B
toAbstract (B t c) = ?????
```

Instead, we will go in the other direction, giving us a :

Add WeChat edu_assist_pro

```
toConc :: [Int] -> Bag
toConc xs = B (sum xs) (length xs)
```

Properties with Refinement Functions

Assignment Project Exam Help

Refinement functions produce properties much like abstraction functions, only with the abstract and concrete layers swapped:

https://eduassistpro.github.io/

```
prop_ref_em
   toConc em ==
```

```
prop_ref_add x ab =
   toConc (addToBagA x ab) == addToBag x (toConc ab)
```

Add WeChat edu_assist_pro

```
prop_ref_avg ab =
   averageBagA ab == averageBag (toConc ab)
```

Assignment 1 and Break

Assignment Project Exam Help

Assignment 1 has b

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Assignment 1 and Break

Assignment Project Exam Help

Assignment 1 has b

It is due right before t

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Assignment 1 and Break

Assignment Project Exam Help

Assignment 1 has b

It is due right before t

https://eduassistpro.github.io/

Advice from Alu

The assignments do not involve much coding, but they do invol

Start early!

Add WeChat edu_assist_pro

Homework

Assignment Project Exam Help

1. Get started o
2. Next progr
3. Last week's
4. This week's quiz is also up, due the following Friday.

https://eduassistpro.github.io/
Add WeChat edu_assist_pro