Assignment Project Exam Help

COMP3141

Software System Design and Implementation

https://powcoder.com

Functors, Applicatives, and Monads Practice

Curtis Millar
CSE, UNSW (and Data61)

15 July 2020

Add WeChat powcoder

Exercise 4

Assignment Project Exam Help

- Capitalise all characters in the input file

https://powcoder.com

Add WeChat powcoder

Exercise 4

Assignment Project Exam Help

- Capitalise all characters in the input file.
- Sum all the numbers in the input file.

https://powcoder.com

Add WeChat powcoder

Exercise 4

Assignment Project Exam Help

- Capitalise all characters in the input file.
- Sum all the numbers in the input file.
- Implement a guessing game AI.

https://powcoder.com

Add WeChat powcoder

State & IO

Assignment Project Exam Help

https://powcoder.com

Week 5 covered State and IO and you've had a few weeks to work with them.
Do you have any questions?

Add WeChat powcoder

Functors, Applicatives, Monads

Assignment Project Exam Help

- Consider *higher-kinded* types of kind `* -> *` that *contain* or *produce* their argument type.

https://powcoder.com

Add WeChat powcoder

Functors, Applicatives, Monads

Assignment Project Exam Help

- Consider *higher-kinded* types of kind `* -> *` that *contain* or *produce* their argument type.
- *Functor* lets us use a pure function to map between the higher-kinded type applied to different concrete types.

https://powcoder.com

Add WeChat powcoder

Functors, Applicatives, Monads

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Consider *higher-kinded* types of kind `* -> *` that *contain* or *produce* their argument type.
- *Functor* lets us use a pure function to map between the higher-kinded type applied to different concrete types.
- *Applicative* lets us apply a *n*-ary function in the context of the higher-kinded type.
- *Monad* lets us *sequentially compose* functions that return values in the higher-kinded type.

Functors

Assignment Project Exam Help

```
class Functor f where
  fmap :: (a -> b) -> f a -> f b
```

The functor typeclass must obey two laws

https://powcoder.com

Add WeChat powcoder

Functors

Assignment Project Exam Help

```
class Functor f where
  fmap :: (a -> b) -> f a -> f b
```

The functor typeclass must obey two laws:

https://powcoder.com

Functor Laws

1. `fmap id == id`
2. `fmap f . fmap g == fmap (f . g)`

Add WeChat powcoder

Applicatives

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
class Functor f => Applicative f where
  pure :: a -> f a
  (<*>) :: f (a -> b) -> f a -> f b
```

The functor type class must obey four additional laws:

Applicatives

Assignment Project Exam Help

```
class Functor f => Applicative f where
  pure :: a -> f a
  (<*>) :: f (a -> b) -> f a -> f b
```

https://powcoder.com

The functor type class must obey four additional laws:

Applicative Laws

1. `pure id <*> v = v` (Identity)
2. `pure f <*> pure x = pure (f x)` (Homomorphism)
3. `u <*> pure y = pure ($ y) <*> u` (Interchange)
4. `pure (.) <*> u <*> v <*> w = u <*> (v <*> w)` (Composition)

Add WeChat powcoder

Alternative Applicative

Assignment Project Exam Help

It is possible to express Applicative equivalently as:

```
class Functor f => App f where
  pure :: a -> f a
  tuple :: f a -> f b -> f (a, b)
```

https://powcoder.com

Example (Alternative Applicative)

Add WeChat powcoder

Alternative Applicative

Assignment Project Exam Help

It is possible to express Applicative equivalently as:

```
class Functor f => App f where
  pure :: a -> f a
  tuple :: f a -> f b -> f (a, b)
```

https://powcoder.com

Example (Alternative Applicative)

1. Using `tuple`, `fmap` and `pure`, let's implement `<*>`.
2. And, using `<*>`, `fmap` and `pure`, let's implement `tuple`.

done in Haskell.

Add WeChat powcoder

Alternative Applicative

Assignment Project Exam Help

It is possible to express Applicative equivalently as:

```
class Functor f => App f where
  pure :: a -> f a
  tuple :: f a -> f b -> f (a,b)
```

https://powcoder.com

Example (Alternative Applicative)

1. Using `tuple`, `fmap` and `pure`, let's implement `<*>`.
2. And, using `<*>`, `fmap` and `pure`, let's implement `tuple`.

done in Haskell.

Add WeChat powcoder

Proof exercise: Prove that `tuple` obeys the applicative laws.

Monads

```
class Applicative m => Monad m where
  (>>=) :: m a -> (a -> m b) -> m b
```

We can define a composition operator with (>>=):

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Monads

Assignment Project Exam Help

```
class Applicative m => Monad m where
  (>>=) :: m a -> (a -> m b) -> m b
```

https://powcoder.com

We can define a composition operator with (>>=):

```
(<=<) :: (b -> m c) -> (a -> m b) -> (a -> m c)
(f <=< g) x = g x >>= f
```

The monad type class must obey three additional laws:

Add WeChat powcoder

Monad Laws

1. `f <=< (g <=< x) == (f <=< g) <=< x` (associativity)
2. `pure <=< f == f` (left identity)
3. `f <=< pure == f` (right identity)

Alternative Monad

Assignment Project Exam Help

It is possible to express Monad equivalently as:

```
class Applicative m => Mon m where
  join :: m (m a) -> m a
```

https://powcoder.com

Example (Alternative Monad)

Add WeChat powcoder

Alternative Monad

Assignment Project Exam Help

It is possible to express Monad equivalently as:

```
class Applicative m => Mon m where
  join :: m (m a) -> m a
```

https://powcoder.com

Example (Alternative Monad)

1. Using `join` and `fmap`, let's implement >>=.
2. And, using >>= let's implement `join`.

Add WeChat powcoder

done in Haskell.

Tree Example

Assignment Project Exam Help

```haskell
data Tree a
  = Leaf
  | Node a (Tree a) (Tree a)
  deriving (Show)
```

https://powcoder.com

Example (Tree Example)

Show that `Tree` is an `Applicative` instance.
done in Haskell!

Add WeChat powcoder

Tree Example

Assignment Project Exam Help

```
data Tree a
   = Leaf
   | Node a (Tree a) (Tree a)
    deriving (Show)
```

https://powcoder.com

Example (Tree Example)

Show that `Tree` is an `Applicative` instance.
done in Haskell!

Add WeChat powcoder

Note that `Tree` is not a `Monad` instance.

Formulas Example

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
data Formula v = Var v
               | And (Formula v) (Formula v)
               | Or  (Formula v) (Formula v)
               | Not (Formula v)
               | Constant Bool
               deriving (Eq,Show)
```

Example (Formulas Example)

Show that `Formula` is a `Monad` instance.

done in Haskell.

Homework

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Homework

Assignment Project Exam Help

1. Week 5's quiz is due on Friday. Make sure you submit your answers.

https://powcoder.com

Add WeChat powcoder

Homework

Assignment Project Exam Help

1. Week 5's quiz is due on Friday. Make sure you submit your answers.
2. The fifth programming exercise is due by the start of my next lecture (in 7 days).

https://powcoder.com

Add WeChat powcoder

Homework

Assignment Project Exam Help

1. Week 5's quiz is due on Friday. Make sure you submit your answers.
2. The fifth programming exercise is due by the start of my next lecture (in 7 days).
3. This week's quiz is also up, it's due Friday week (in 9 days).

https://powcoder.com

Add WeChat powcoder

Consultations

Assignment Project Exam Help

- Consultations will be made on request. Ask on piazza or email `cs3141@cse.unsw.edu.au`.

https://powcoder.com

Add WeChat powcoder

Consultations

Assignment Project Exam Help

- Consultations will be made on request. Ask on piazza or email `cs3141@cse.unsw.edu.au`.
- If there is a consultation it will be announced on Piazza with a link a room number for Hopper.

https://powcoder.com

Add WeChat powcoder

Consultations

Assignment Project Exam Help

- Consultations will be made on request. Ask on piazza or email `cs3141@cse.unsw.edu.au`.
- If there is a consultation it will be announced on Piazza with a link a room number for Hopper.
- Will be in the Thursday lecture slot, 9am to 11am on Blackboard Collaborate.

https://powcoder.com

Add WeChat powcoder

Consultations

Assignment Project Exam Help

- Consultations will be made on request. Ask on piazza or email `cs3141@cse.unsw.edu.au`.
- If there is a consultation it will be announced on Piazza with a link a room number for Hopper.
- Will be in the Thursday lecture slot, 9am to 11am on Blackboard Collaborate.
- Make sure to join the queue on Hopper. Be ready to share your screen with REPL (`ghci` or `stack repl`) and editor set up.

https://powcoder.com

Add WeChat powcoder