Quiz (Week 2)

Types and Constructors

Consider the following type definitions.

```
type B = Bool
data X = A X
       | B B
       | C Y
newtype Y = Y B
```

Question 1

Which of the following can identifiers can stand for *types* in the above definitions? Check all that apply.

1. ☐ `A`
2. ☐ `B`
3. ☐ `C`
4. ☐ `X`
5. ☐ `Y`

Question 2

Which of the following identifiers can stand for *constructors* in the above definitions? Check all that apply.

1. ☐ `A`
2. ☐ `B`
3. ☐ `C`
4. ☐ `X`
5. ☐ `Y`

Types in Design

For each of the following use cases, choose the type definitions that best reflect this use case, eliminating as many possible errors or invalid values as possible.

Question 3

A connection is in one of three states: *connected*, *connecting* or *disconnected*. It also contains the following information:

- The destination IP address
- If it is in the *disconnected* state, the time it disconnected.
- If it is in the *connected* state, the arrival time of the most recent packet, if one exists.
- If it is in the *connected* state, the size of the most recent packet, if one exists.
- If it is in the *connecting* state, the time the connection was initiated.

1. ○

```
data State = Connected | Connecting | Disconnected

data Connection
  = Connection
      { destination      :: IPAddress
      , state            :: State
      , timeDisconnected :: Maybe Time
      , timeInitiated    :: Maybe Time
      , packetArrival    :: Maybe Time
      , packetSize       :: Maybe Size
      }
```

2. ○

```
data State
  = Connected    { packetArrival :: Maybe Time
                 , packetSize    :: Maybe Size
                 }
  | Connecting   { timeInitiated    :: Time }
  | Disconnected { timeDisconnected :: Time }

data Connection
  = Connection
      { destination :: IPAddress
      , state       :: State
      }
```

3.

```haskell
data State = Connected | Connecting | Disconnected

data Connection
  = Connection
      { destination      :: IPAddress
      , state            :: State
      , timeDisconnected :: Maybe Time
      , timeInitiated    :: Maybe Time
      , recentPacket     :: Maybe (Time, Size)
      }
```

4.

```haskell
data State
  = Connected    { recentPacket     :: Maybe (Time, Size) }
  | Connecting   { timeInitiated    :: Time }
  | Disconnected { timeDisconnected :: Time }

data Connection
  = Connection
      { destination :: IPAddress
      , state       :: State
      }
```

Question 4

A message is encrypted using a password. The system will not allow messages to be encrypted with weak passwords. Messages can only be logged if encrypted.

1.

```haskell
checkStrength :: String -> Bool
encrypt       :: String -> String -> String
log           :: String -> Log -> Log
```

2.

```haskell
newtype Encrypted = E String
checkStrength :: String -> Bool
encrypt       :: String -> String -> Encrypted
log           :: Encrypted -> Log -> Log
```

3.

```
newtype Password  = P String
newtype Encrypted = E String
checkStrength :: String -> Maybe Password
encrypt       :: String -> Password -> Encrypted
log           :: Encrypted -> Log -> Log
```

4. ◯

```
newtype Password  = P String
checkStrength :: String -> Maybe Password
encrypt       :: String -> Password -> String
log           :: String -> Log -> Log
```

Monoids and Semigroups

Question 5

Which of the following `Semigroup` instances are *lawful*?

1. ☐

```
newtype X = X Bool
instance Semigroup X where
  X a <> X b = X (not a || b)
```

2. ☐

```
newtype X = X Bool
instance Semigroup X where
  X a <> X b = X (a || b)
```

3. ☐

```
newtype X = X Bool
instance Semigroup X where
  X a <> X b = X (a `xor` b)
    where xor True  x = not x
          xor False x = x
```

4. ☐

```
newtype X = X Bool
instance Semigroup X where
  X a <> X b = X a
```

Question 6

Which of the following is a valid *monoid*?

1. ☐ The type `Integer`, the `max` function and identity element `0`
2. ☐ The type `Bool`, the `(&&)` function and identity element `False`
3. ☐ The type `Bool`, the `(&&)` function and identity element `True`
4. ☐ The type `Integer`, the function `(\a b -> (a + b) `div` 2)`, and identity element `0`.

Relations

Question 7

Check all of the following that are valid *partial orders*.

1. ☐ `(==)`
2. ☐ `\x y -> x `mod` y == 0`
3. ☐ `(>=)`
4. ☐ `(/=)`

Question 8

Check all of the following that are valid *equivalence relations*.

1. ☐ `(==)`
2. ☐ `\x y -> x `mod` 10 == y `mod` 10`
3. ☐ `(>=)`
4. ☐ `(/=)`

Due: Friday, June 21, 11:59:59 pm

Upon clicking submit, you will be prompted for your zID and zPass. Please make sure that your answers are final and that you have answered every question.

If there is a problem, please contact the course administrator.

Submit

You can click here to check if you have submitted already.