Dashboard / My courses / FP (2022/2023 - R.R. Slavescu) / 19 December - 25 December / Haskell test - Labs 8 - 10 (30433/1)

Started on	Tuesday, 20 December 2022, 10:05 AM
State	Finished
Completed on	Tuesday, 20 December 2022, 10:12 AM
Time taken	6 mins 44 secs
Grade	**5.17** out of 10.00 (**52**%)

Question **1**

Incorrect

Mark 0.00 out of 1.00

Select the snippets that are valid Haskell code

☑ a. ✖
```
len l = case l of
   [] -> 0
   (_::xs) -> 1 + len xs
```

☑ b. ✔
```
inc :: Num a => a -> a
inc a = a + 1
```

☑ c. ✖
```
inc : Num a => a -> a
inc a = a + 1
```

☑ d. ✔
```
len l = case l of
   [] -> 0
   (_:xs) -> 1 + len xs
```

Your answer is incorrect.

Question **2**

Incorrect

Mark 0.00 out of 1.00

Given the following function definition:

```
f :: [Int] -> Int
f [1, 2] = 1
f [_, _] = 2
f [3, 4] = 3
```

the result of the following function call is:

```
f [3, 4]
```

Answer: 3 ✖

Question **3**

Partially correct

Mark 0.67 out of 1.00

Which function describes best the each of the following list comprehensions?

[x^2 | x <- xs]	map	✓
[take 3 | x <- xs]	take	✗
[x | x <- xs, x `div` 3 == 2]	filter	✓

Your answer is partially correct.

You have correctly selected 2.

Question **4**

Correct

Mark 1.00 out of 1.00

Which of the following are examples of **valid** ways to create local definitions in Haskell?

☑ a. let y = 5 in y * 1 ✓

☐ b. y * 2 with y = 5

☑ c. y * 2 where y = 5 ✓

☐ d. local y = 5 in y * 2

Your answer is correct.

Question **5**

Correct

Mark 1.00 out of 1.00

Select the function that uses pattern guards correctly to implement the filter function:

● a. ✓

```
filter _ [] = []
filter p (x:xs)
  | p x = x:filter p xs
  | otherwise = filter p xs
```

○ b.

```
filter _ [] = []
filter p (x:xs)
  | p x -> x:filter p xs
  | else -> filter p xs
```

○ c.

```
filter _ [] = []
filter p (x:xs) =
  if p x then x:filter p xs
  otherwise filter p xs
```

Your answer is correct.

Question **6**

Correct

Mark 1.00 out of 1.00

Select all the **true** statements about the bottom value:

☐ a. In Haskell, Nothing is the bottom value

☑ b. In Haskell, undefined is the bottom value ✓

☑ c. The bottom value can be assigned to any type ✓

☐ d. The compiler won't compile (i.e. will show an error) programs that contain the bottom value

Your answer is correct.

Question **7**

Incorrect

Mark 0.00 out of 1.00

Select all the **true** statements about type classes

☑ a. All type class implementations for a data type must be in the module where the data is defined ✗

☐ b. Type classes are used to define classes, types that also have methods and private fields

☑ c. We can implement type classes defined by the standard library for our own types ✓

☐ d. Type classes are used to abstract common behavior for various types (like Java interfaces)

Your answer is incorrect.

Question **8**

Correct

Mark 1.00 out of 1.00

The following list comprehension:

```
[(x, y) | x <- ['a', 'b']; y <- [1, 2]]
```

○ a. Returns [('a', 1), ('b', 2)]

○ b. Returns [('a',1),('a',2),('b',1),('b',2)]

● c. Fails to combile because the syntax is invalid ✓

○ d. Fails to combile because x and y have different types

Your answer is correct.

Question **9**

Incorrect

Mark 0.00 out of 1.00

Select the correct functions such that the definition of m3 below multiplies 3 numbers wrapped in Maybe

mul3 a b c = a * b * c

m3 a b c = `` `fmap` `` `mul3` a `<*>` b `<*>` c

Your answer is incorrect.

Question **10**

Partially correct

Mark 0.50 out of 1.00

Given the following code:

```
newtype All = All Bool
instance Semigroup All where
   (All a) <> (All b) = All (a && b)
instance Monoid All where
   mempty = All True
```

The result of the following expressions is:

```
foldl (<>) mempty (map (\x -> All (x >= 2)) [2, 3, 4])
```

All True ✓

```
foldl (<>) mempty (map (\x -> All (x `mod` 2 == 0)) [1, 2, 3])
```

All True ✗

Your answer is partially correct.

You have correctly selected 1.

◄ Haskell test - Labs 8 - 10 (30433/2)

Jump to...

L12-Synopsis-Lazy-evaluation ►