Started on	Tuesday, 22 November 2022, 12:15 PM
State	Finished
Completed on	Tuesday, 22 November 2022, 12:30 PM
Time taken	14 mins 49 secs
Grade	**7.08** out of 10.00 (**71**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Please select True for the Default mark.

Select one:

- True ✔
- False

Information

The next 5 questions are basic questions.

Read each question carefully.

Question **2**

Correct

Mark 1.00 out of 1.00

The following code snippet:

```
view = div [] [style "color" "red", text "Some text"]
```

a. Will render the text "Some text" with black color, because the style is not applied correctly

b. Will render the text "Some text" with red color

c. Will generate invalid HTML that causes the browser to show an error

d. Will fail to compile ✔

Your answer is correct.

Question **3**

Correct

Mark 0.50 out of 0.50

Select the components of the Elm architecture

- [x] a. Update ✔
- [x] b. View ✔
- [x] c. Model ✔
- [ ] d. Action
- [ ] e. Component

Your answer is correct.

Question **4**

Correct

Mark 1.00 out of 1.00

The function countVowels can be rewritten using pipelines as:

countVowels s = List.length (List.filter isVowel (List.map Char.toLower s))

- [ ] a. countVowels s = s |> List.length |> List.filter isVowel |> List.map Char.toLower
- [ ] b. countVowels s = List.map Char.toLower <| List.filter isVowel <| List.length <| s
- [x] c. countVowels s = List.length <| List.filter isVowel <| List.map Char.toLower <| s ✔
- [x] d. countVowels s = s |> List.map Char.toLower |> List.filter isVowel |> List.length ✔

Your answer is correct.

Question **5**

Partially correct

Mark 0.83 out of 1.00

Function composition operator >> takes as first parameter a function and second parameter a function and returns a value .

The pipeline operator |> takes as parameter first parameter a value and second parameter a function and returns a value .

Note: first parameter is on the left hand side of the operator and second parameter is on the right hand side of the operator.

Your answer is partially correct.

Question **6**

Partially correct

Mark 0.25 out of 0.50

To get the value that is inside the Just variant of Maybe or provide a default value, we can:

- a. Use the Maybe.withDefault function
- b. Use a case expression ✔
- c. Use an if expression
- d. Use the Maybe.unwrap function

Your answer is partially correct.

You have correctly selected 1.

Information

The next 3 questions are intermediate questions.

Read each question carefully.

Question **7**

Correct

Mark 1.00 out of 1.00

Select the correct way(s) of creating input fields using Elm.

- [ ] a. input ("text" "email" email EmailChanged) []
- [x] b. input [type_ "text", placeholder "email", value email, onInput EmailChanged] [] ✔
- [x] c. inputAttrs ty p v msg = [type_ ty, placeholder p, value v, onInput msg] ✔
  input (inputAttrs "text" "email" email EmailChanged) []
- [ ] d. input ["text" "email" email EmailChanged] []

Your answer is correct.

Question **8**

Incorrect

Mark 0.00 out of 1.00

Given the following definitions:

```
inc x = x + 1
dec x = x - 1
double x = x * 2
```

What does the expression below evaluate to?

```
((inc >> double) << (double << (double >> dec))) 3
```

Answer: 26 22 ✖

Question **9**

Correct

Mark 1.00 out of 1.00

Given the following definitions:

```
type DivError = DivByZero
```

```
divNums : Int -> Int -> Result DivError Int
divNums a b =
  if a == 0 then
    Err DivByZero
  else
    Ok (b // a)
```

The type of the following expressions is:

divNums 2 10 |> Result.mapError (_ -> "Division by zero!")	Result String Int ✔
divNums 2 10 |> Result.andThen (divNums 2)	Result DivError Int ✔
divNums 2 10	Result DivError Int ✔

Your answer is correct.

Information

The next 2 questions are advanced questions.

Read each question carefully.

Question **10**

Not answered

Marked out of 1.00

The result of the following expression is:

```
type alias Point = {x: Int, y: Int}

points = [{x = 3, y = 1}, {x = 3, y = 2}, {x = 3, y = 5}]

mx : Int -> List Point -> List Point
mx d ps = ps |> List.map (\p -> { p | x = p.x * d - 2 })

my : Int -> List Point -> List Point
my d ps = ps |> List.map (\p -> { p | y = p.y * d - 1 })

points |> mx 1 |> my 2 |> List.map .y |> List.foldl (+) 0
```

Answer: 13 ✖

Question **11**

Partially correct

Mark 0.50 out of 1.00

Select all the **true** statements:

- [x] a. We can use the same accessor to access fields that have the same name but are from different record types ✔
- [ ] b. Trying to view the type of an accessor in the REPL will result in an error, because they have a special type
- [ ] c. Records use nominal typing
- [ ] d. To pass an accessor to map or filter, we use the .accessor syntax

Your answer is partially correct.

You have correctly selected 1.

◄ L08-Code

Jump to...

L09-Synopsis ►