Time left 0:05:42

Given the following combinator:

```
manyTill :: Parser a -> Parser b -> Parser [b]
manyTill end content = orElse pEnd pContent where
   pEnd = pMap (const []) end
   pContent = pMap (\(x,xs) -> x:xs) (andThen s (manyTill end s))
```

That will keep parsing `content` until the `end` parser succeeds.
This happens by first trying the end parser, if it succeeds, the parsing ends, otherwise the content parser is tried and the result is added to a list.

Given:

```
result = Success ("123","")
```

and

```
input = "\"a1b2c3\""
```

Select the parser definition that satisfies `runParser p input == result`

Hint: Try to find a pattern in the input and connect that with the output before considering the parser definitions below!

a. `p = (char '"') `pThen` (manyTill digit (char '"'))`

b. `p = (char '"') `andThen` (manyTill alpha (char '"')) where alpha = letter `orElse` digit`

c. `p = manyTill alpha (char '"') where alpha = letter `orElse` digit`

d. `p = (char '"') `pThen` (manyTill (lower `pThen` digit) (char '"'))`

Clear my choice

Nu bag mana in foc

Complete the parser below such that it parses a C\C++ array containing numbers (e.g. {1,2,3}):

cArray = between (char '{') (char '}') $ (sepBy (char ',') number)

Back

Time le

Question **4**
Answer saved
Marked out of 1.00
Flag question

Statements like `putStrLn "Please enter your name:"` are desugared to the `>>` function.

Bindings like `name <- getLine` are desugared to the `>>=` function.

Tir

Question **2**

Answer saved

Marked out of 1.00

Flag question

Given the following parser:

```
p = number `andThen` many (pThen (char ',') number)
```

Select the inputs that will successfully parse (i.e. will yield Success _).

Note: the parser doesn't have to consume all of the input in order to yield the Success variant!

Hint: Try to express in words (natural language) what the parser does before considering the inputs below.

- [x] a. 123,4,5,
- [ ] b. 123,4,5,abc
- [ ] c. 1,2,3
- [ ] d. 1

Back

Question **6**

Answer saved

Marked out of 1.00

Flag question

Select the correct functions such that the definition of m2 below adds 2 numbers wrapped in Maybe

m2 a b = (+) <$> a <*> b

Back

Question **10**

Answer saved

Marked out of 1.00

Flag question

Select the **false** statements about monads in Haskell:

- [ ] a. Monads allow us to write imperative looking code using do notation
- [x] b. Monoid is an alias Monad
- [x] c. Monad defines the function <*>
- [ ] d. Maybe is an example of a Monad

Previous page

Back

Question **9**

Answer saved

Marked out of 1.00

Flag question

Select the function that is the equivalent of the following function written in do notation

```
fn = do
   [src, dst] <- getArgs
   contents <- readFile src
   writeFile dst contents
```

a. `getArgs >>= \[src,dst] -> readFile src >> writeFile dst`

b. `getArgs >>= \[src,dst]-> readFile src >>= writeFile dst`

c. `getArgs >> \[src,dst] -> readFile src >> writeFile dst`

d. `getArgs >>= readFile >>= writeFile`

Clear my choice

Question **7**

Answer saved

Marked out of 1.00

Flag question

Select the **false** statements about monads in Haskell:

- [x] a. Monad defines the function <$>
- [ ] b. Monad is a type class
- [x] c. Int is an example of Monad
- [ ] d. Maybe is an example of a Monad

Back

Question **5**

Answer saved

Marked out of 1.00

Flag question

Select all the **false** statements about Input/Output in Haskell

- [ ] a. Haskell's *main* function has the signature *main :: IO ()*
- [ ] b. To obtain a line from the standard input, we can write

```
do
  name <- getLine
  putStrLn name
```

- [x] c. To read data from a file we use the *read* function
- [x] d. do notation can be only used with the IO monad

Previous page

Back

Question **8**

Answer saved

Marked out of 1.00

Flag question

Select the **true** statements about monads in Haskell:

- [x] a. Monad defines the function >>=
- [x] b. [a] (List) is an example of a Monad
- [ ] c. Monad is an alias for Monoid
- [ ] d. Int is an example of a Monad

Previous page