Time left 0:11:14

6 7

Question **1**

Not yet answered

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.

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

Next page

Time le

Question **2**

Not yet answered

Marked out of 1.00

Flag question

Given the following combinator:

```
rep :: Int -> Parser a -> Parser [a]
rep 0 p = succeed []
rep n p = pMap (\(a, as) -> a:as) $ andThen p (rep (n-1) p)
```

That applies a given parser a fixed number of times and returns the results in a list.

Select the parser definition that would yield:

```
Success ("123","")
```

for the following input:

```
AA1BB2CC3
```

i.e. 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 = rep 3 (andThen (rep 2 upper) digit)
- b. p = rep 3 (pThen (pThen upper upper) digit)
- c. p = rep 3 (andThen (pThen upper upper) digit)
- d.
```
p = andThen s (andThen s s) where
    ld = (pThen (pThen upper upper) digit)
```

ion 4
et
ered
ed out of
g question

Select the function signature that **best** represents a parser

- a. String -> a
- b. String -> Result ParseError (a, String)
- c. [Int] -> Result Int a
- d. String -> Result ParseError a

5

d

out of

question

Which of the following names would best describe the following parser:

```
satisfies (`elem` ['0'..'9'])
```

- a. upper
- b. lower
- c. digit
- d. char

Question **3**

Not yet answered

Marked out of 1.00

Flag question

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

```
fn = do
   putStrLn "Line to reverse"
   line <- getLine
   putStrLn (reverse line)
```

>>= așteaptă func cu un arg

>> 2 monazi, return the second while still running the first one

a. `putStrLn "Line to reverse" >> getLine >>= \line -> putStrLn (reverse line)`

b. `putStrLn "Line to reverse" >>= getLine >>= \line -> putStrLn (reverse line)`

c. `putStrLn "Line to reverse" >> getLine >> \line -> putStrLn (reverse line)`

d. `putStrLn "Line to reverse" >>= getLine >>= putStrLn (reverse line)`

Time left 0:04:06

Question 7

Not yet answered

Marked out of .00

Flag question

Given the following combinator:

```
rep :: Int -> Parser a -> Parser [a]
rep 0 p = succeed []
rep n p = pMap (\(a, as) -> a:as) $ andThen p (rep (n-1) p)
```

That applies a given parser a fixed number of times and returns the results in a list.

Select the parser definition that would yield:

```
Success ("ab","3")
```

for the following input:

```
ab123
```

i.e. 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 = pThen (pThen lower lower) (pThen digit digit)
- b. p = pMap (\(a, b) -> a ++ b) $ andThen (rep 2 lower) (rep 2 digit)
- c. p = pMap fst $ andThen (rep 2 lower) (rep 2 digit)
- d. p = pThen (rep 2 lower) (rep 2 digit)

stion 8

yet
wered

ked out of

ag question

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

- [x] a. To obtain a line from the standard input, we can write

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

- [x] b. Haskell's *main* function has the signature *main :: IO ()*
- [ ] c. To read data from a file we use the *read* function
- [ ] d. do notation can be only used with the IO monad

cmd

readFile for files
works with every monad

evious page