Laboratorul 13: Model Examen - Definirea monadei 'IO' folosind monada 'State'

Înainte de a rezolva acest exercițiu ar fi util sa va reamintiți modul în care am definit propria monada IO în cursul 10:

- ne-am definit propria monadă `MyIO`

```
type Input = String
type Output = String

newtype MyIO a =
        MyIO { runMyIO :: Input -> (a, Input, Output)}

instance Monad MyIO where
   return x = MyIO (\input -> (x, input, ""))
   m >>= k  = MyIO f
            where f input =
              let (x, inputx, outputx) = runMyIO m input
                  (y, inputy, outputy) = runMyIO (k x) inputx
              in  (y, inputy, outputx ++ outputy)

instance Applicative MyIO where
   pure      = return
   mf <*> ma = do { f <- mf; a <- ma; return (f a) }

instance Functor MyIO where
   fmap f ma = do { a <- ma; return (f a) }
```

- am definit operațiile de bază

```
myPutChar :: Char -> MyIO ()
myPutChar c = MyIO (\input -> ((), input, [c]))

myGetChar :: MyIO Char
myGetChar = MyIO (\ (c:input) -> (c, input, ""))

runIO :: MyIO () -> String -> String
runIO command input = third (runMyIO command input)
```

```
                    where third (_, _, x) = x
```

- folosind operațiile de bază am definit mai multe operații derivate: `myPutStr`, `myPutStrLn`, `myGetLine`, `echo`
- am făcut legătura cu monada `IO` folosind funcția

```
convert :: MyIO () -> IO ()
convert = interact . runIO
```

Model EXAMEN: `MyIOState`

Definiți propria monada `IO` folosind monada `State`, plecând de la următoarele definiții:

```
type Input = String
type Output = String
type InOutWorld = (Input, Output)
type MyIOState a = State InOutWorld a
```

Atenție! acestă definiție trebuie scrisă într-un fișier care conține definiția monadei `State`.

- înțelegeți cum arată o valoare de tipul `MyIOState`
- definiți operațiile de bază `myGetChar` , `myPutChar` și `runIO`
- definiți operații derivate utile (`myPutStr`, `myGetLine`)
- testați definițiile (puteți folosi exemplele din cursul 10)