|  |
| --- |
| module Task4\_1 where |
|  |  |
|  | -- Монада над функцией. В качестве входного значения `fun` может быть что угодно |
|  | -- Собственно, почему бы не `String`? |
|  | data FunMonad a = FunMonad { fun :: String -> a } |
|  |  |
|  | -- реализуйте классы `Functor`, `Applicative` и `Monad` для типа `FunMonad` |

instance Functor FunMonad where

fmap f (FunMonad x) = FunMonad (\() -> f (x ()))

instance Applicative FunMonad where

pure x = FunMonad $ \() -> x

FunMonad a <\*> FunMonad b = FunMonad $ \() -> (a ()) (b ())

instance Monad FunMonad where

return a = FunMonad (\() -> a)

m >>= k = k (fun m ())

|  |
| --- |
| module Task4\_2 where |
|  |  |
|  | data FourOf a = FourOf a a a a deriving(Show,Eq) |
|  |  |
|  | -- реализуйте классы `Functor`, `Applicative` и `Monad` для типа `FourOf` |
|  | -- таким образом, что |
|  | -- do { x <- FourOf 1 2 3 4; y <- FourOf 4 6 7 8; return $ x + y } === FourOf 5 8 10 12  instance Functor fourof where  Fmap f ( fourof a a a a ) = fourof ( f a a a a )  Instance applicative fourof where  Pure = fourof  ( fourof f ) <\*> (fourof a a a a ) = fourof ( f a a a a )  Instance monad fourof where  Return x = fourof x  ( fourof x ) >> f = f x |