Examen Programare Declarativă. 10 februarie 2017

Subiectul 1 (6 puncte). Se dă următoarea secvență de declarații în Haskell:

Barem:
0.4 Lit + <= & Not
0.5 If & Var
1 p. Alegerea monadei Reader (Pentru State, doar 0.75) (otherwise 0.4)
0.4 evalAExp :: AExp -> Reader Env Int
0.4 evalBExp :: AExp -> Reader Env Int
0.4 Do
0.4 Return
0.4 lookup folosind get / definitia monadei

```
type Id = String
data AExp = Lit Int | Var Id | AExp :+: AExp | If BExp AExp AExp
data BExp = AExp :<=: AExp | BExp :&: BExp | Not BExp
type Env = [(Id, Int)]
eval :: AExp -> Env -> Int
```

Să se completeze definiția funcției `eval` pentru a putea calcula valoarea unei expresii dat fiind un "mediu de evaluare" (`Env`) care asociază întregi numelor de variabile. Observație: puteți folosi mai multe funcții ajutătoare.

Dacă în timpul evaluării un nume de variabilă nu este găsit, se poate presupune că valoarea acesteia este 0.

Punctaj: 6 puncte din care 3 puncte rezolvarea, + 2 rezolvarea folosind monade + 1 rezolvarea folosind monada **predefinită** cea mai potrivită.

Subiectul 2 (3 puncte). Se dă următoarea secvență de declarații în Haskell:

```
class Armata soldat where
  recrutare :: major -> soldat major
  antrenament :: (neinstruit -> soldat instruit)
                                  -> (soldat neinstruit -> soldat instruit)

-- 1 punct
armeaza :: (Armata soldat) => (neinstruit -> instruit)
                                  -> (soldat neinstruit -> soldat instruit)
-- 2 puncte
inmulteste :: (Armata soldat) => (neinstruit -> soldat instruit)
                                     -> ([neinstruit] -> soldat [instruit])
```

Să se definească cele două funcții de mai sus în modul cel mai natural posibil.

Punctaj: jumatate din punctaj pentru orice definitie care se compileaza; punctaj intreg pentru o definitie naturala care ia in calcul toate argumentele de intrare.

1 punct din oficiu