Exemplele sunt la final.

Se dă tipul de date: data Exp = Var String -- Variabilă x

| Val Int -- Întreg 5

| Op Exp String Exp -- Operație 3 <= x

| If Exp Exp Exp -- Conditional if (3 <= x) then 3 else x

| Lambda String Exp -- Funcție anonimă \ x -> 3 + x

**Cerințe:**

1. Să se scrie o funcție subExps care ia ca argument o expresie și întoarce lista subexpresiilor ei directe.  
   subExps :: Exp -> [Exp]  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Să se scrie o funcție aggExps care ia ca argument o expresie e și o listă de expresii es și înlocuiește subexpresiile directe ale lui e cu cele din es, în ordine, dacă nr. subexpresiilor coincide cu lungimea lui es; dacă nu, expresia rămâne neschimbată. (aggExps e un fel de inversă a lui subExps)  
   aggExps :: Exp -> [Exp] -> Exp  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Să se scrie o funcție foldExp care ia ca argument o funcție (de agregare) f și o expresie e și calculează o valoare astfel: calculează recursiv valorile corespunzătoare subexpresiilor directe ale lui e, apoi cheamă funcția f cu expresia e și lista de valori obținută pentru a obține valoarea finală. Sugestie: folosiți funcția subExps.  
   foldExp :: (Exp -> [b] -> b) -> Exp -> b  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. Să se scrie o funcție transform care ia ca argument o funcție t de transformare (locală) și o expresie e și întoarce expresia obținută prin aplicarea funcției t fiecărei subexpresii a lui e, de jos în sus. Sugestie: folosiți foldExp și aggExps  
   transform :: (Exp -> Exp) -> Exp -> Exp  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Să se scrie o funcție freeVars care ia ca argument o expresie e și calculează lista variabilelor libere care apar în e. Pentru toți constructorii se reunesc variabilele din subexpresii, cu excepția lui Lambda pentru care se elimină variabila definită. Sugestie: folosiți foldExp și o funcție de agregare nouă.  
   freeVars :: Exp -> [String]  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

subExps (Op (Op (Val 3)"+"(Var "x"))"\*"(Var "x")) = [Op (Val 3)"+"(Var "x"),Var "x"]

aggExps (Op (Var "x") "+" (Val 3)) [Val 2,Val 3] = Op (Val 2) "+" (Val 3)

aggExps (Op (Var "x") "+" (Val 3)) [Val 2] = Op (Var "x") "+" (Val 3)

foldExp (\ \_ c -> 1 + sum c) (Op (Op (Val 3) "+" (Var "x")) "\*" (Var "x")) = 5

transform (\ e -> case e of { Var "x" -> Val 2; \_ -> e}) (Op (Op (Val 3) "+" (Var "x")) "\*" (Var "x")) = Op (Op (Val 3) "+" (Val 2)) "\*" (Val 2)

freeVars (Op (Lambda "x" (Var "x")) "" (Var "y")) = ["y"]

freeVars (Op (Lambda "x" (Var "y")) "" (Var "x")) = ["y","x"]