Programare Funcțională
Strategii de evaluare.

În acest laborator, vom face experimente cu strategii de evaluare utilizate în lambda-calcul.

Exercițiul 0.1. Reamintim funcțiile `reduce1` și `reduce1'`:

```
data Term = Var Id
     | App Term Term
     | Lambda Id Term deriving (Show, Eq)
reduce1' :: Term -> [Id] -> Maybe Term
reduce1' (Var id') _ = Nothing
reduce1' (App (Lambda id term) term') avoid =
  Just (casubst id term' term avoid)
reduce1' (App term1 term2) avoid = case reduce1' term1 avoid of
  Nothing -> case reduce1' term2 avoid of
               Nothing -> Nothing
               Just term2' -> Just (App term1 term2')
  Just term1' -> Just (App term1' term2)
reduce1' (Lambda id term) avoid = case reduce1' term avoid of
  Nothing -> Nothing
  Just term' -> Just (Lambda id term')

reduce1 :: Term -> Maybe Term
reduce1 t = reduce1' t (vars t)
```

Întrebare: Care este strategia de evaluare implementată mai sus? Testați strategia pe exemplele următoare:

$$1.(\lambda x_1.x_1)\ \Big((\lambda x_2.x_2)\ (\lambda z.(\lambda x_3.x_3)\ z)\Big);$$
$$2.(\lambda x_1.\lambda x_2.x_2)\ \Big((\lambda x.x)\ (\lambda y.y)\Big).$$

Exercițiul 0.2. Pornind de la implementarea strategiei de la Exercițiul 0.1, implementați strategia *call-by-name* (CBN). Reamintim că pentru această strategie nu sunt permise reducerile în interiorul unei lambda-abstracții. Testați implementarea strategiei peste exemplele de la Exercițiul 0.1. Puteți identifica un avantaj al strategiei CBN?

Testați strategia și pe următorul exemplu:

$$(\lambda x_1.x_1\ x_1)\ \Big((\lambda x.x)\ (\lambda y.y)\Big).$$

Câți pași are calculul asociat acestei strategii?

Exercițiul 0.3. Care este strategia de evaluare implementată de funcțiile de mai jos?

```
strategy1' :: Term -> [Id] -> [Term]
strategy1' (Var _) _ = []
strategy1' (App (Lambda id term) term') avoid = [casubst id term' term avoid] ++
  let all = strategy1' term avoid in
  let all' = strategy1' term' avoid in
  [ App (Lambda id successorTerm) term' | successorTerm <- all ] ++
  [ App (Lambda id term) successorTerm' | successorTerm' <- all']
strategy1' (App term1 term2) avoid =
  let all1 = strategy1' term1 avoid in
  let all2 = strategy1' term2 avoid in
  [ App sterm1 term2 | sterm1 <- all1 ] ++
  [ App term1 sterm2 | sterm2 <- all2 ]
strategy1' (Lambda id term) avoid =
  let all = strategy1' term avoid in
  [ Lambda id sterm | sterm <- all ]

strategy1 :: Term -> [Term]
strategy1 term = strategy1' term (vars term)

strategy :: Term -> [Term]
strategy term = let all = strategy1 term in case all of
   [] -> [term]
   _ -> concat (map strategy all)
```

Testați această strategie peste exemplele din exercițiile anterioare.

Exercițiul 0.4. Implementați strategia *call-by-value* (CBV) și testați implementarea pe exemplele din exercițiile anterioare.

Exercițiul 0.5. Comparați calculele asociate strategiilor CBV și CBN atunci când ele sunt rulate peste exemplele:

$$1.(\lambda x_1.\lambda x_2.x_2)\ \Big((\lambda x.x)\ (\lambda y.y)\Big);$$
$$2.(\lambda x_1.x_1\ x_1)\ \Big((\lambda x.x)\ (\lambda y.y)\Big).$$

Exercițiul 0.6. Implementați strategia Applicative Order.