Programare Funcțională
Lambda calcul simplu tipizat.
Laborator

În acest laborator, vom începe implementarea unui interpretor pentru lambda calcul simplu tipizat. Mai departe, vom utiliza termenul de lambda expresie pentru a ne referi strict la expresii lambda simplu tipizate.

Exercițiul 0.1. Utilizând notele de curs, completați definiția Haskell de mai jos cu sintaxa pentru lambda calcul tipizat. Care sunt diferențele față de lambda calcul netipizat?

```
{- tip de date pentru tipurile simple -}
data Type = TyBool
          | ...

{- identificatorii/variabilele din lambda-calcul -}
type Id = String

{- lambda-termenii -}
data Term = Var Id
          | Lam Id Type Term
          | App Term Term
          | ...
          | ...
          | ...
```

Exercițiul 0.2. Scrieți o valoare de tip Term care reprezintă lambda-termenul $(\lambda x : Bool.x)true$.

Mai departe, vom utiliza tipul de date `Context` corespunzător contextelor de tipizare. Reamintim că un context de tipizare Γ conține perechi de forma $x : T$, unde x este un identificator și T este un tip. Funcția `update` actualizează un context *Gamma* cu o nouă pereche $x : T$.

```
{- tip de date pentru contextul de tipare -}
```

```
data Context = Empty -- contextul vid
             | Bind Id Type Context

-- adauga un binding intr-un context
update :: Context -> Id -> Type -> Context
update gamma x ty = Bind x ty gamma
```

Exercițiul 0.3. Scrieți o funcție care caută tipul unei variabile într-un context.

```
search :: Context -> Id -> Maybe Type
search Empty _ = Nothing
search (Bind x ty gamma) y = ...
```

Reamintim că un raționament de tip (engleză *typing judgement*) este de forma $\Gamma \vdash t : T$ (se citește: t are tipul T în contextul Γ) și poate fi construit folosind următoarele reguli de tipizare:

$$\text{T-True}\ \frac{}{\Gamma \vdash \mathit{true} : \mathit{Bool}} \qquad \text{T-False}\ \frac{}{\Gamma \vdash \mathit{false} : \mathit{Bool}}$$

$$\text{T-App}\ \frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2} \qquad \text{T-Var}\ \frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

$$\text{T-Lam}\ \frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x : T_1 . t : T_1 \to T_2} \qquad \text{T-If}\ \frac{\Gamma \vdash t_1 : \mathit{Bool} \qquad \Gamma \vdash t_2 : T \qquad \Gamma \vdash t_3 : T}{\Gamma \vdash \mathit{if}\ t_1\ \mathit{then}\ t_2\ \mathit{else}\ t_3 : T}$$

Exercițiul 0.4. Completați funcția de mai jos care are rolul de a verifica dacă un lambda termen t este are tipul T sau nu. Funcția va întoarce `Nothing` dacă nu există T astfel încât $\Gamma \vdash t : T$ sau JustT, dacă $\Gamma \vdash t : T$.

Pentru fiecare caz identificați regula de tipizare corespunzătoare.

```
typecheck :: Context -> Term -> Maybe Type
typecheck gamma (Var id) = ...
typecheck gamma TTrue = Just ...
typecheck gamma TFalse = Just ...
typecheck gamma (TIte t1 t2 t3) =
  case typecheck gamma t1 of
    Nothing -> Nothing
    Just TyBool -> let tyt2 = typecheck gamma t2 in
                   let tyt3 = typecheck gamma t3 in
```

```
                    if tyt2 == tyt3 then
                      ...
                    else
                      Nothing
    Just _ -> ...
typecheck gamma (App t1 t2) =
  case typecheck gamma t1 of
    Just (TyArrow ty1 ty2) -> case typecheck gamma t2 of
                                ...
      _ -> Nothing
    _ -> Nothing
typecheck gamma (Lam x ty t) =
  case typecheck (update gamma x ty) t of
    Just ty' -> ...
    _ -> Nothing
```

Exemplu de utilizare:

```
{-

*Main> t1
App (Lam "x" TyBool (Var "x")) TTrue
*Main> typecheck Empty t1
Just TyBool

-}
```

Exercițiul 0.5. Testați funcția de mai sus pentru:

- celelalte exemple din curs;
- un lambda-termen care nu este bine tipizat;
- un lambda-termen cu variabile libere și un context care nu conține acele variabile.

Exercițiul 0.6. Adaptați funcția `reduce` (și celelalte funcții necesare) din laboratorul precedent.

Exercițiul 0.7. Adăugați tipul `TyNat` și construcțiile aferente din curs.