Zadanie 1.

(A) Podaj przykład termu który dla którego strategia gorliwej ewaluacji doprowadza do postaci normalnej w mniejszej liczbie redukcji niż strategia normalizująca (leftmost outermost).
(B) Podaj przykład termu który dla którego strategia normalizująca doprowadza do postaci normalnej w mniejszej liczbie redukcji niż strategia gorliwa.
Dla obu przykładów podaj liczbę redukcji prowadzącą do postaci normalnej w obu strategiach.

Zadanie 2 Napisz drzewo wywodu typu

$$(((o- > o)- > o)- > o)- > o- > o$$

w systemie typów prostych dla poniższego termu (w pustym kontekście).

$$\lambda a\ b.a(\lambda x.(x\ b))$$

Zadanie 3 Przypisz poniższym termom najbardziej ogólne typy w systemie Hindley-Milner w kontekście który liście pustej `[]` przypisuje polimorficzny typ `'a list`.

```
letrec
    fun S a b c = K a b c (K b a c)
    fun K a b = a
in S K K end
```

```
let
    fun K a b = a
in let
    fun S a b c = K a b c (K b a c)
    in S K K end
end
```

```
let
    fun K a b = a
in let
    fun L x y = S y x []
    fun S a b c = a c (b c)
    in S K K end
end
```

Dla przypomnienia reguły podstawowe reguły typowania :

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \quad [\texttt{Var}]$$

$$\frac{\Gamma \vdash e_0 : \tau \to \tau' \qquad \Gamma \vdash e_1 : \tau}{\Gamma \vdash e_0\ e_1 : \tau'} \quad [\texttt{App}]$$

$$\frac{\Gamma,\ x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda\ x\ .\ e : \tau \to \tau'} \quad [\texttt{Abs}]$$

$$\frac{\Gamma \vdash e_0 : \sigma \qquad \Gamma,\ x : \sigma \vdash e_1 : \tau}{\Gamma \vdash \texttt{let}\ x = e_0\ \texttt{in}\ e_1 : \tau} \quad [\texttt{Let}]$$

$$\frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma} \quad [\texttt{Inst}]$$

$$\frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{free}(\Gamma)}{\Gamma \vdash e : \forall\ \alpha\ .\ \sigma} \quad [\texttt{Gen}]$$

(τ, τ' oznaczają zmienne typowe, σ, σ' schematy typów)
(aby otrzymać pełny zestaw reguł trzeba jeszcze uwzględnić `letrec` wraz z wielokrotnymi definicjami)

Zadanie 4 Zaproponuj implementację poniższego wariantu katenowalnych list tak aby podane operacje działały w podanych amortyzowanych czasach (n oznacza liczbę elementów na liście). Amortyzowane czasy powinny pozostać tego samego rzędu również w przypadku presystentego użycia. W rozwiązaniu można wykorzystać strumienie oraz notację dla uleniwiania używane na wykładzie.

```
signature CList = sig
    type 'a clist

    val fromList :'a list -> 'a clist  (* O(1) *)
    val toList   :'a clist -> 'a list  (* O(n) *)
    val cat      :'a clist * 'a clist -> 'a clist (* O(1) *)
end
```

Zadanie 5 Przepisz poniższą implementację liczb tak aby operacje miały amortyzowane czasy odpowiednio `inc,dec`-$O(1)$ oraz `toInteger`-$O(\log n)$ (n jest wartością liczby reprezentowanej przez strukturę). Amortyzowane czasy powinny pozostać tego samego rzędu również w przypadku presystentego użycia. Zaproponuj funkcję potencjału (długu struktury), która dowodzi że amortyzowane czasy są odpowiednie. W rozwiązaniu można wykorzystać strumienie oraz notację dla uleniwiania używane na wykładzie.

```
datatype Digit= ONE | TWO | THREE
type Number = Digit list

fun inc [] = [ONE] |
        inc (ONE::xs)   = (TWO::xs) |
        inc (TWO::xs)   = (THREE::xs) |
        inc (THREE::xs) = (TWO::inc xs)

fun dec (THREE::xs) = (TWO::xs) |
        dec (TWO::xs)   = (ONE::xs) |
        dec [ONE]       = []         |
        dec (ONE::xs)   = (TWO::dec xs)

fun toInteger ds = let
    fun dI ONE   = 1 |
        dI TWO   = 2 |
        dI THREE = 3
    fun tI [] mult acc = acc |
        tI (digit::ds) mult acc = tI ds (2*mult) (acc+ mult*(dI digit))
    in tI ds 1 0 end
```