Zadanie 1. Udowodnij Lema Newmana:

Lemat. 1 *Jeśli relacja ma słabą własność Churcha-Rossera (WCR) oraz własność silnej normalizacji (SN) to ma własność Churcha-Rossera (CR).*

WCR – jeśli $b \leftarrow a \rightarrow c$ to $b \twoheadrightarrow d \twoheadleftarrow c$ (dla pewnego d)
CR – jeśli $b \twoheadleftarrow a \twoheadrightarrow c$ to $b \twoheadrightarrow d \twoheadleftarrow c$ (dla pewnego d)
SN – nie istnieje nieskończony ciąg $a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow \ldots$

Zadanie 2

Poniższy λ-term zredukuj na trzy sposoby za pomocą redukcji gorliwej, leniwej oraz *normal order reduction.*

$$(\lambda sz.s(s(z)))(\lambda x.SKKx)$$

(gdzie $K = \lambda ab.a$ oraz $S = \lambda abc.(ac)(bc)$)

Zadanie 3

Znajdź najbardziej ogólny typ w systemie Hindley-Milner dla poniższego termu. Rozbijmy klauzulę `letrec` używając analizy zależności. Czy typ nowo powstałego termu jest inny, jeśli tak to jaki?

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

Dla przypomnienia reguły podstawowe reguły typowania :

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

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

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

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

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

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

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

Zadanie 4

Przekształć poniższy program do postaci superkombinatorów:

```
letrec
    sumints= \m. letrec
        count = \m. IF (> n m ) NIL (CONS n (count (+ n 1)))
        in sum (count 1)
    sum= \ns. IF (= ns NIL) 0 (+ (HEAD ns) (sum (TAIL ns)))
in sumInts 100
```

Zadanie 5 Uzupełnij implementację funkcji `check` w poniższej implementacji kolejki. Operacje kolejki powinny działać w amortyzowanym czasie stałym (również w przypadku persystentnego użycia). Uzasadnij że dostarczona implementacja spełnia ten wrunek.

```
type a Queue = a list * int * a list susp * int * a list

val empty = ([],0,$[],0,[])
fun isEmpty (_,lenf,_,_,_) = (lenf = 0)

fun check (q as (w,lenf,f,lenr,r)) =
```

```
fun snoc ((w,lenf,f,lenr,r), x) = check (w,lenf,f,lenr+1,x::r)
fun head (x::w,lenf,f,lenr,r) = x
fun tail (x::w,lenf,f,lenr,r) = check (w,lenf-1,$tl (force f),lenr,r)
```