IB015 Zápočet, skupina E 60 minut **Datum: 8. 1. 2019**

Jméno: **UČO:** **Souřadnice:**

0007 *list* *učo* *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0123456789

Pro udělení zápočtu je třeba vyřešit správně všech pět příkladů v této písemce. Jedna drobná chyba bude tolerována.

1. [Haskell] Naprogramujte funkci `myZipWith :: (a → b → c) → [a] → [b] → [c]`, která se chová stejně jako její obdoba ze standardní knihovny `zipWith`. Tato funkce vrací seznam hodnot, které jsou výsledkem aplikace funkce zadané jako první parametr na odpovídající hodnoty seznamů zadaných jako druhý a třetí parametr. Nezapomeňte, že výsledný seznam je nejvýše tak dlouhý jako kratší ze zadaných seznamů.

Můžete využít libovolné konstrukce jazyka Haskell, nesmíte však použít knihovní funkci `zipWith`.

Příklady vyhodnocení:

```
myZipWith (+) [3,2,1] [1,2,3,4] ⇝* [4,4,4]
myZipWith (\_ a → a) [1,2,3] "ab" ⇝* "ab"
myZipWith (,) [] [1,2,3] ⇝* []
```

```
myZipWith _ [] _ = []
myZipWith _ _ [] = []
myZipWith f (x:xs) (y:ys) = (f x y) : (myZipWith f xs ys)
```

2. [Haskell] Pro následující výraz programovacího jazyka Haskell uveďte, čím nahradit text ??? tak, aby platilo uvedené vyhodnocení.

```
??? odd [7..11] ⇝* [True, False, True, False, True]
```

```
??? → map
```

Oblast strojově snímaných informací, nezasahujte. ***Druhá strana se neskenuje.***

Jméno: | UČO: | Souřadnice:

0007 *list* 2 *učo* *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0 1 2 3 4 5 6 7 8 9

3. [Haskell] Mějme dán datový typ `ExpM`, jehož hodnoty slouží k uložení aritmetického výrazu tvořeného celočíselnými hodnotami a libovolnou typově korektní aplikací binárního odčítání.

```
data ExpM = Con Integer
          | Diff ExpM ExpM
```

a) Uveďte hodnotu typu `ExpM`, která odpovídá výrazu `((3-4)-5)`.

b) Napište funkci `eval :: ExpM → Integer`, která pro libovolnou hodnotu typu `ExpM` vypočte skutečnou hodnotu přidruženého aritmetického výrazu.

a) Diff (Diff (Con 3) (Con 4)) (Con 5)

b)
```
eval :: ExpM -> Integer
eval (Con x) = x
eval (Diff x y) = ((eval x) - (eval y))
```

IB015 Zápočet, skupina E 60 minut **Datum: 8. 1. 2019**

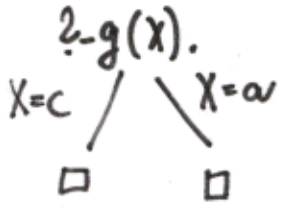
Jméno: **UČO:** **Souřadnice:**

0007 *list* 3 *učo* *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0123456789

4. [Prolog] Vypište ve správném pořadí všechny možné odpovědi (uživatel vkládá středník), které interpret Prologu vrátí na dotaz `?- g(X).` za předpokladu následující databáze pravidel a faktů.

```
f(X, Y) :- g(X), h(X, Y).
h(c, Y) :- g(X).
g(c).
g(a).
```



?- g(X).
X=c ;
X=a.

Oblast strojově snímaných informací, nezasahujte. ***Druhá strana se neskenuje.***

IB015 Zápočet, skupina E 60 minut **Datum: 8. 1. 2019**

Jméno: **UČO:** **Souřadnice:**

0007 *list* 4 *učo* *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0123456789

5. [Prolog] Do programovacího systému Prolog převeďte níže uvedenou funkci `foo` programovacího jazyka Haskell, tj. definujte predikát `myFoo(+XS,?YS)`, který uspěje právě pro takové dvojice argumentů, pro které platí, že aplikace `foo` na první argument vrátí hodnotu druhého argumentu. Můžete předpokládat, že seznam uvedený jako první argument je neprázdný seznam čísel.

Funkce v Haskellu:

```
foo :: [Integer] → Integer
foo xs = last xs + 42
```

```
myFoo(+XS,?YS)
myFoo([X],Y) :- Y is X+42.
myFoo([_|XS],Y) :- myFoo(XS,Y).
```