

Jméno:

UČO:

Souřadnice:

0007

list

učo

body

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

0123456789

Základní část

Příklady v této části písemky ověřují základní znalosti. Vyřešení všech příkladů v této části je nutné pro úspěšné ukončení předmětu. Tato část písemky se nebuduje.

1. **[povinné][Haskell]** Naprogramujte v jazyce Haskell funkci `myZip :: [a] → [b] → [(a,b)]`, která se chová stejně jako její obdoba `zip` ze standardní knihovny. Tato funkce spojí dva seznamy hodnot do jednoho seznamu uspořádaných dvojic hodnot. Můžete využít libovolné konstrukce jazyka Haskell, nesmíte však použít knihovní funkce `zip` a `zipWith`.

Příklady vyhodnocení:

```
myZip [1, 2, 3] [1, 2, 3] ~>* [(1,1),(2,2),(3,3)]
```

```
myZip "abc" "efghij" ~>* [('a','e'),('b','f'),('c','g')]
```

```
myZip [1,2] [] ~>* []
```

2. **[povinné][Haskell]** Uveďte typ výrazu `uncurry (^)` v jazyce Haskell. Typy elementárních výrazů jsou:

```
uncurry :: (a → b → c) → (a, b) → c
```

```
(^) :: (Integral b, Num a) ⇒ a → b → a
```

Jméno:

UČO:

Souřadnice:

0007

list

2

učo

body

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

0123456789

3. [povinné][Haskell] Je dán datový typ `BinTree` a reprezentující v Haskellu binární stromy, které v listech mají uložené seznamy hodnot typu `a`.

```
data BinTree a = Leaf [a] | Node (BinTree a) (BinTree a)
```

Definujte funkci `double` (včetně jejího typu), která pro zadaný strom seznamů výše uvedeného typu vrátí strom zdvojených seznamů, tj. všechny seznamy přidružené ke všem listům původního stromu zdvojí.

Příklady vyhodnocení:

```
double (Leaf [1,2]) ~>* Leaf [1,2,1,2]
```

```
double (Node (Leaf "abc") (Leaf "d")) ~>* Node (Leaf "abcabc") (Leaf "dd")
```

Jméno:

UČO:

Souřadnice:

0007

list

3

učo

body

Oblast strojově snímaných informací. Svě ucho a číslo listu vyplňte
zleva dle vzoru číslic. Jinak do této oblasti nezasahujte.

0123456789

4. [povinné][Prolog] Zakreslete kompletní SLD strom pro následující dotaz a databázi faktů.

?- u(X, a).

z(a).

z(c).

zz(a).

u(X, Y) :- zz(X), Y = b.

u(X, Y) :- z(X), z(Y).

Jméno:

UČO:

Souřadnice:

0007

list

4

učo

body

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

0123456789

5. [povinné][Prolog] Naprogramujte v jazyce Prolog predikát `only(+List, ?X)`, který uspěje právě tehdy, pokud je seznam `List` tvořen pouze prvky `X`. Predikát musí fungovat i pro neinstanciované `X`, predikát nemusí být definován pro prázdný seznam a můžete předpokládat, že `List` bude seznam čísel. Všechna řešení, která predikát vrátí, musí být správná, až na případné `false`. za poslední správnou odpovědí.

Příklady vyhodnocení:

```
?- only([2,2,2], 2).  
true .
```

```
?- only([1,1,1], X).  
X = 1 .
```

```
?- only([1,2], X).  
false.
```

Jméno:

UČO:

Souřadnice:

0007

líst

5

učo

body

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

0123456789

Pokročilá část

Příklady v této části písemky jsou bodované. Body, které získáte spolu s body za domácí úlohy a aktivitu na cvičeních, rozhodnou o Vaší známce. Tato část písemky Vám bude opravena pouze pokud úspěšně vyřešíte základní část.

6. [5 bodů][Haskell] Uvedte přesný typ výrazu ($\backslash x \rightarrow \text{dropWhile } x [1, 2, 3]$) v jazyce Haskell. Typy elementárních výrazů jsou:

```
dropWhile :: (a -> Bool) -> [a] -> [a]
```

```
1 :: Num a => a
```

```
2 :: Num a => a
```

```
3 :: Num a => a
```

7. [10 bodů][Haskell] Naprogramujte v Haskellu funkci `sublistMax :: Ord a => [[a]] -> [a]`, která pro daný seznam seznamů vrátí ten z těchto seznamů, který obsahuje největší prvek. Nemusíte uvažovat případy, kdy vnější, ani kterýkoli z vnitřních seznamů bude prázdný. Pokud dva vnitřní seznamy obsahují stejný maximální prvek, tak je jedno, který z nich vrátíte.

Příklad použití:

```
sublistMax [[3,2,1], [1,4]] ~>* [1, 4]
```

Jméno:

UČO:

Souřadnice:

0007

list

6

učo

body

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

0123456789

8. [10 bodů][Haskell] Naprogramujte v jazyce Haskell funkci `main :: IO ()`, která provádí následující vstupně-výstupní akci:

1. požádá uživatele smysluplným výpisem aby napsal jméno souboru;
2. načte ze vstupu řetězec (jméno souboru);
3. pokud je řetězec prázdný, ukončí se, jinak vypíše na výstup obsah zadaného souboru a opakuje celý postup.

Nemusíte řešit co se stane, pokud některý ze souborů nebude existovat. Můžete využít například následující funkce:

```
readFile :: FilePath → IO String
```

```
getLine :: IO String
```

```
putStrLn :: String → IO ()
```

(FilePath je alias pro String).

Jméno:

UČO:

Souřadnice:

0007

list

7

učo

body

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

0123456789

9. [10 bodů][Prolog] Zakreslete výpočetní SLD strom následujícího programu pro dotaz $?-f(X, Y)$. Zakreslete také, kde se nacházejí větve, kterými výpočet nepokračuje, a popište, o které použití řezu se jedná (ořezání/upnutí).

a(1).

a(2).

f(X, Y) :- g(X, Y).

f(3, Y) :- a(Y).

g(X, Y) :- a(X), !, a(Y).

g(1, 1).