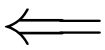
Písomná skúška FP: 29. mája 2024

Táto písomná skúška trvá 90 minút. Počas skúšky je povolené používať laptop s inštalovaným kompilátorom jazyka Haskell ghc, resp. ghci, cabal a/alebo stack, Visual Studio Code a/alebo emacs a/alebo IntelliJ IDEA. Iné používateľské aplikácie, najmä komunikačné aplikácie, webové prehliadače a aplikácie pre synchronizáciu súborov, nie sú povolené. Knihy, dokumentácia v papierovej podobe, písané poznámky atď. nie sú povolené.

Túto písomnú skúšku je potrebné vypracovať samostatne, teda bez pomoci niekoho iného a bez komunikácie s niekým iným. Odhalené podvádzanie pri skúške, napr. kopírovanie odpovedí a riešení (aj ich častí), snaha odovzdať cudzie odpovede a riešenia a pod., môže byť penalizované vylúčením študenta z predmetu, nepridelením a/alebo zrušením bodov, prípadne aj disciplinárnym konaním v zmysle Študijného poriadku, ktoré môže viesť k vylúčeniu zo štúdia.

Meno a priezvisko: 

---

1. Prepíš funkciu *myreverse* pomocou rekurzívneho volania. (5 b.)

```
1 myreverse :: Foldable t => t a -> [a]
2 myreverse = foldl (flip (:)) []
```

Riešenie:

```
1 myreverse' xs = reverseHelper xs []
2
3 reverseHelper :: [a] -> [a] -> [a]
4 reverseHelper [] acc = acc
5 reverseHelper ... = ...
```

---

2. Ktorá funkcia z *Data.List* je striktná v prvom argumente, ale lenivá v druhom argumente? Ako je v *ghci* možné ukázať, že druhý argument je naozaj lenivý? (5 b.)

Riešenie:

3. Naimplementuj funkciu $maximum$ pomocou $foldr$. Funkcia $maximum$ vráti prvok s najvyššou hodnotou. Hint: riešenie je na 1 riadok. (5 b.)

```
maximum :: (Foldable t, Ord a) => t a -> a
```

Riešenie:

---

4. Volanie $zipWith$ je typovo nesprávne, pretože funkcia $addPair$ očakáva dvojicu hodnôt. Čo je potrebné pridať do volania $zipWith$, aby toto volanie vygenerovalo výstup $[5, 7, 9]$? Akým spôsobom to vyrieši tento problém? Požiadavka: pridajte 1 špecifickú funkciu. (5 b.)

```
addPair :: (Int, Int) -> Int
addPair (x, y) = x + y

zipWith (addPair) [1, 2, 3] [4, 5, 6]
```

Riešenie:

---

5. Aký je rozdiel medzi $foldl$ a $foldl'$? Popíš aj kľúčové implementačné rozdiely. (5 b.)

```
foldl :: Foldable t => (b -> a -> b) -> b -> t a -> b
foldl' :: Foldable t => (b -> a -> b) -> b -> t a -> b
```

Riešenie:

6. Uveď príklad adhoc polymorfizmu v jazyku Haskell. Prečo je potrebný? (5 b.)

Riešenie:

---

7. Prečo je volanie *three infinity* rozdielne vyhodnotené v prípade okamžitej a lenivej evaluácie? (5 b.)

```
1 infinity :: Integer
2 infinity = 1 + infinity
3
4 three :: Integer -> Integer
5 three x = 3
6
7 three infinity
```

Riešenie:

8. Aký je rozdiel medzi operátormi >>= a >>? Ako sa tento rozdiel odzrkadľuje na typovej signatúre týchto operátorov? (5 b.)

```
1  (>>=) :: Monad m => m a -> (a -> m b) -> m b
2  (>>) :: Monad m => m a -> m b -> m b
```

Riešenie:

---

9. Pomocou množinovej abstrakcie vyfiltruj dvojice s hodnotou *Nothing* zo zoznamu. (10 b.)

```
ls :: Num a => [(a, Maybe a)]
ls = [(0, Just 1), (1, Just 8), (2, Nothing), (3, Just 10)]  — príklad zoznamu
```

Riešenie: