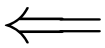
Písomná skúška FP: 13. júna 2022

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, a to aj bodov z cvičení, prípadne aj disciplinárnym konaním v zmysle Študijného poriadku, ktoré môže viesť k vylúčeniu zo štúdia.

Meno a priezvisko: 

Študentské ID:

---

1. Naimplementujte funkciu, ktorá inkrementuje hodnoty vo vnorených zoznamoch, napr. `incr [[1,2,3], [4,5,6]]` **vráti** `[[2,3,4], [5,6,7]]`**. Funkciu** `incr` **naimplementujte na 1 riadok, bez použitia** `do`**. (4 b.)**

Riešenie:

```
1 incr :: [[Int]] -> [[Int]]
```

---

2. Naimplementujte funkciu `foo` **pomocou rekurzívneho volania. Funkcia** `foo` **spojí dva zoznamy, ktoré obsahujú neklesajúce hodnoty, a vráti zoznam neklesajúcich hodnôt, napr.** `foo [1,3,5] [1,2,4,6]` **vráti** `[1,1,2,3,4,5,6]`**,** `foo [] []` **vráti** `[]`**. (4 b.)**

Riešenie:

```
1 foo:: [Int] -> [Int] -> [Int]
```

---

3. Naimplementujte funkciu `foo` **pomocou rekurzívneho volania. Funkcia** `foo` **pre zoznam neklesajúcich hodnôt vyfiltruje opakujúce sa hodnoty, napr.** `foo [1,1,1,2]` **vráti** `[1,2]`**,** `foo []` **vráti** `[]`**. (4 b.)**

Riešenie:

```
1 foo :: [Int] -> [Int]
```

4. Prečo je možné použiť funkciu (+), ktorá má typ `Num a => a -> a -> a`, teda očakáva použitie numerického typu `Num` a nie typu `Maybe Int`? (4 b.)

```
1 bar :: Maybe Int
2 bar = (+) <$> Just 1 <*> Just 2
```

Riešenie:

---

5. Prepíšte kód pomocou `fmap`, bez použitia `<$>`. Prečo je to možné? (4 b.)

```
1 bar :: Maybe Int
2 bar = (+ 1) <$> Just 1
```

Riešenie:

---

6. Funkcia (+) je strict pre jeden argument, pre oba argumenty, alebo pre žiaden argument? Prečo? (4 b.)

Riešenie:

7. Prečo je vrátená hodnota `8`? Prečo nie je `undefined` nikdy vyhodnotené? (4 b.)

```
foldl (\x y -> y) undefined [undefined, 8]
```

Riešenie:

---

8. Prečo potrebujeme `deriving (Show)`? (4 b.)

```
data Employee = Employee
  { name :: String,
    age :: Int
  }
  deriving (Show)
```

Riešenie:

---

9. Prepíšte funkciu `foo` bez použitia `>>=`! (4 b.)

```
foo :: Maybe Int -> Maybe Int -> Maybe Int
foo x y = x >>= (\a -> y >>= (\b -> return $ a + b))
```

Riešenie:

10. Funkciu `map` je možné prepísať pomocou `foldr` (pozri príklad dole). Prečo nie je možné použiť `foldl` namiesto `foldr`? (4 b.)

```
1  f :: [Int] -> [Int]
2  f xs = map (+ 1) xs
```

```
1  f' :: [Int] -> [Int]
2  f' xs = foldr (\x acc -> (x + 1) : acc) [] xs
```

Riešenie:

---

11. Prepíšte kód bez použitia Bang patterns (do riešenia stačí uviesť modifikovaný riadok 7)! Je potrebná aj strict evaluácia `x`? (5 b.)

```
1  {-# LANGUAGE BangPatterns #-}
2
3  fac :: (Integral a) => a -> a
4  fac x = fac' x 1
5    where
6      fac' 1 y = y
7      fac' x !y = fac' (x - 1) (x * y)
```

Riešenie:

---

12. Výsledok evaluácie je `[[3,2,3],[3,5,6],[5,4,4]]`. Ako je potrebné doplniť kód tak, aby vypočítal maximum v každom podzozname, teda v našom prípade by výsledok bol `[3, 6, 5]` (max. prvok v `[3,2,3]` je `3`, max. prvok v `[3,5,6]` je `6` atď.)? Nápoveda: použite `$`. (5 b.)

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

Riešenie: