Zadania z funkcionálneho programovania

1 Kategória

1. Definujte funkciu *numbersCount* pre zistenie, počtu číslic v reťazci.

```
numbersCount :: Num a => [Char] -> a
```

2. Definujte funkciu $transform$ pre transformáciu čísla z dvojkovej sústavy do desiatkovej.

```
transform :: [Char] -> Int
```

3. Definujte funkciu *atoi* pre transformáciu reťazca na číslo. Číslo môže začínať znamienkom +, alebo -.

```
atoi :: [Char] -> Int
```

4. Definujte 2 funkcie jedného argumentu (reťazca), ktoré vrátia dvojicu reťazcov, pričom prvý reťazec nech obsahuje len malé písmená z pôvodného reťazca a druhý reťazec nech obsahuje všetky ostatné znaky.
   Nech funkcia $smallAndOther1$ je definovaná prostredníctvom funkcie $filter$.
   Nech funkcia $smallAndOther2$ je definovaná pomocou množinovej abstrakcie.

```
smallAndOther1, smallAndOther2 :: [Char] -> ([Char],[Char])
```

5. Definujte 2 funkcie jedného argumentu (zoznamu celých čísel), ktoré vrátia zoznam prvočísel zo zoznamu argumentu.
   Nech funkcia $primeFilter1$ je definovaná prostredníctvom funkcie $filter$.
   Nech funkcia $primeFilter2$ je definovaná pomocou množinovej abstrakcie.

```
primeFilter1, primeFilter2 :: [Int] -> [Int]
```

6. Definujte funkciu *primeGenerator* jedného argumentu n, ktorá vygeneruje zoznam prvých n prvočísel.

```
primeGenerator :: Int -> [Int]
```

7. Definujte funkciu *intToBin* jedného argumentu (celého čísla), ktorá vypočíta jeho binárnu hodnotu. Binárna hodnota nech je reprezentovaná ako zoznam cifier, pričom poradie cifier nech je zľava doprava. V prípade, že argument funkcie *intToBin* nie je kladné číslo, funkcia nech vráti prázdny zoznam.
   Definujte funkciu *binToString* pre transformáciu zoznamu čísel na reťazec prostredníctvom kompozície funkcií *concat*, *map* a *intToBin*.

```
intToBin ::  Int -> [Int]
binToString :: [Int] -> [Char]
```

8. Definujte funkciu *htoi*, ktorá vypočíta hodnotu celého šestnástkového čísla z reťazca číslic. Použite pri tom funkciu $foldl$.

```
htoi :: [Char] -> Int
```

9. Na vstupe nech je zoznam kladných čísel rozdelených do skupín oddelených číslom 0. Napr. zoznam $[1, 2, 5, 0, 3, 6, 8, 2]$ obsahuje dve skupiny $[1, 2, 5]$ a $[3, 6, 8, 2]$.
   Definujte funkciu, ktorá z takého zoznamu vypočíta zoznam obsahujúci súčty čísel v každej skupine (pre uvedený príklad bude výsledok $[8, 19]$). Použite funkciu $foldl$.

```
groups :: [Int] -> [Int]
```

2 Kategória

1. Definujte funkciu *mergeSort* pre triedenie zlučovaním (merge sort), ktorá bude funkciu pre zlučovanie očakávať ako prvý parameter.
   Vytvorte dve funkcie pre zlučovanie, ktoré usporiadajú prvky zostupne, resp. vzostupne.

```
mergeSort :: ([a]->[a]->[a]) -> [a] -> [a]

mergeDesc :: Ord a => [a] -> [a] -> [a]
mergeAsc  :: Ord a => [a] -> [a] -> [a]
```

2. Definujte funkciu *insertSort* pre usporiadanie prvkov zoznamu priamym vkladaním (insert sort).

```
insertSort ::  Ord a => [a] -> [a]
```

3. Definujte funkciu *selectSort* pre usporiadanie prvkov zoznamu priamym výberom (select sort).

```
selectSort ::  Ord a => [a] -> [a]
```

4. Definujte funkciu *shellSort* pre usporiadanie prvkov zoznamu podľa algoritmu shell sort.

```
shellSort :: Ord a => [a] -> [a]
```

5. Definujte funkciu *bubbleSort* pre usporiadanie prvkov zoznamu podľa algoritmu bubble sort.

```
bubbleSort :: Ord a => [a] -> [a]
```

6. Definujte funkciu, ktorá nakreslí diamant so zadanou dĺžkou uhlopriečky. Napr.:

```
Main> diamant 5
  *
 ***
*****
 ***
  *
```

7. Definujte funkciu, ktorá realizuje prihlasovanie používateľa do systému. Nech funkcia ako parametre prijíma databázu používateľov, meno a heslo. Heslá sú v databáze ukladané v šifrovanej podobe.

```
type HashedPassword = Int
type PassDB = [(String, HashedPassword)]

login :: PassDB -> String -> String -> Bool
```

Definujte tiež funkciu pre pridanie používateľa do databázy.

```
addPassword :: (String,String) -> PassDB -> PassDB
```

Extra kategória

1. Vytvorte program pre zobrazenie čísla (aj viaccifferného) pomocou sedem segmentoviek. Každá cifra vo vstupnom čísle bude zobrazená jednou sedem segmentovkou.

   Jednou sedem segmentovkou nech je možné zobraziť 10 obrazcov. Každý obrazec nech je reprezentovaný jednocifernou číslicou. Predpokladom je, že každý obrazec zodpovedá tvaru číslice, ktorá ho reprezentuje.

   Konfigurovateľnosť obrazcov nech je jednoduchá. Každý obrazec nech je reprezentovaný zoznamom celých čísel, kde tieto číslice indexujú jednotlivé segmenty sedem segmentovky. Napr. ak sa budú indexovať segmenty zhora dole a zľava doprava, potom obrazce čísel môžu byť definované nasledovne:

```
cisla :: [[Int]]
cisla = [[1,2,3,5,6,7],[3,6],[1,3,4,5,7],[1,3,4,6,7],[2,4,6],[1,2,4,6,7],
         [2,4,5,6,7],[1,3,6],[1,2,3,4,5,6,7],[1,2,3,4,6]]
```

Je potrebné realizovať funkciu *printDigit* n s = ..., kde:
n - číslo, ktoré je potrebné zobraziť
s - šírka sedem segmentovky

Napr. pri aplikácii *printDigit* 1234567890 7 nech je výstup v tvare:

```
   *****   *****           *****           *****   *****   *****   *****
*       *       * *       *       *               * *     * *     * *     *
*       *       * *       *       *               * *     * *     * *     *
*       *       * *       *       *               * *     * *     * *     *
*       *       * *       *       *               * *     * *     * *     *
*       *       * *       *       *               * *     * *     * *     *
   *****   *****   *****   *****   *****           *****   *****
* *             *       *       * *     *       * *     *       * *     *
* *             *       *       * *     *       * *     *       * *     *
* *             *       *       * *     *       * *     *       * *     *
* *             *       *       * *     *       * *     *       * *     *
* *             *       *       * *     *       * *     *       * *     *
   *****   *****           *****   *****           *****           *****
```

2. Nech je definovaný typ *Symbol*, ktorý určuje jeden zo symbolov v hre Piškvorky (X,O) alebo prázdne pole (N).

```
data Symbol = X | O | N
    deriving (Eq, Show)
```

Definujte funkciu *checkWinner*, ktorá zistí, kto je výhercom hry. Funkcia vráti symbol používaný výhercom (X alebo O) alebo N v prípade, že nikto nevyhral. Za výhercu sa považuje ten, kto má v rade za sebou stanovený počet svojich symbolov.

Parametrami funkcie nech sú počet symbolov v rade potrebný na výhru a hracie pole (ako dvojrozmerný zoznam symbolov). Pole môže mať ľubovoľnú veľkosť, ale je vždy štvorcové.

```
checkWinner :: Int -> [[Symbol]] -> Symbol
```

3. Definujte funkciu pre overenie správnosti formátu zadania rodného čísla. Výsledkom nech je zoznam hodnôt typu *Bool*, ktorý hovorí, či zadané rodné číslo vyhovuje resp. nevyhovuje definovanému formátu.

Formát rodného čísla nech je v tvare: $XXXXXX[<separator>]XXX[X]$, kde X je číslica a "*separator*" môže byť ľubovoľný viacznakový oddeľovač v rodnom čísle, napr. "/","−","− >",(žiaden oddeľovač). Definujte funkciu, ktorá určí zoznam povolených separátorov nasledovne:

```
separator = ["","/","-","->"]
```

Definujte nasledujúce typové synonymum pre typ rodného čísla:

```
type IDNumber = String
```

Nech vstupná databáza rodných čísel určených na verifikáciu je nasledujúca:

```
persons :: [IDNumber]
persons = ["850228/7772","846224/7772","846224777","12345->123",
           "850228->6666","123456->>123","880230-8889"]
```

Potom výstup bude v tvare:

```
[True,True,True,False,True,False,True]
```

Program ďalej rozšírte o validáciu hodnôt zadaných rodných čísel s určeným formátovaným výstupom, kde pre určený vstup definovaný funkciou *persons* bude výstup v tvare:

```
28.2.1985...........Man
24.12.1984..........Woman
24.12.1984..........Woman -> validation error (postfix = 777)
28.2.1985...........Man -> validation error (postfix = 6666)
30.2.1988...........Man -> validation error (postfix = 8889)
```

Nech sa zobrazia sa iba tie rodné čísla, ktoré majú správny formát. Počet znakov v riadku pre zobrazenie dátumu nasledovanými znakmi '.' nech je konfigurovateľný prostredníctvom nasledujúcej funkcie:

```
lineLength :: Int
lineLength = 40
```

Údajové typy nech sú definované nasledovne:

```
data Sex = Man | Woman deriving(Eq,Show)

type IDNumber = String
type Postfix = String {- ID number postfix -}
type Year = Int
type Month = Int
type Day = Int
type Person =
  (Year,Month,Day,Sex,Postfix) {- Persons data extracted from ID number -}
```

Pre overenie rodného čísla platí nasledujúca podmienka:

```
(den+mesiac+rok+postfix) `mod` 11 == 0
```

Samozrejme overte, či je mesiac v rozmedzí povolených hodnôt a taktiež či deň je v rozmedzí povolených hodnôt vo vzťahu na rok a mesiac. Napr. formát 880230/8889 nech nie je akceptovaný.