Funkcionális programozás / mintatétel

1. Egészítsük ki az alábbi függvényt úgy, hogy az meghatározza a paraméterként megadott `nr` szám `p` számrendszerbeli számjegyei összegét. Adjuk meg a függvény típusdefinícióját, egy kezdeti kifejezést és a kiértékelés eredményét.

```
sDigit p nr
   | nr < p = ...
   | otherwise = ...
         where
         temp = ...
         digit = nr `rem` p
```

2. Mit csinál az alábbi függvény, ha a függvény paramétere 5? Adjuk meg a függvény típusdefinícióját, egy kezdeti kifejezést és a kiértékelés eredményét.

```
szamit n = map (\x -> x `rem` 2) [1..n]

A. [1, 0, 1, 0, 1]
B. [True,False,True,False,True]
C. [True,True,False,False,True]
D. Komplilálási hiba
```

3.Mit csinál az alábbi `fofugv` függvény? Adjuk meg a megfelelő típusdefiníciókat, majd egy kezdeti kifejezést és a kiértékelés eredményét.

```
fofugv = fugv (\x -> x `rem` 2 == 0) [1,3,4,7,8,-5,6,-4,7,8]

fugv f [] = []
fugv f (k: ve)
   |f k = (k: fugv f ve)
   |otherwise = fugv f ve
```

4. Keressük meg a hibákat az alábbi függvényben, feltételezve, hogy a `sort` beszúrásos rendezéssel rendezi a paraméterként megadott lista elemeit. Adjuk meg a típusdefiníciókat és egy kezdeti kifejezést.

```
sort [] = []
sort (x: xs) = sSort x xs

sSort e [] = [e]
sSort e (x: xs)
   | e > x = (x: sSort xs)
   | otherwise = (e: (x: xs))
```

5. Egészítsük ki, az alábbi `fugv` és `fo` függvényeket, úgy hogy a `fo` függvény bináris kereséssel határozza meg, hogy hányadik helyen szerepel a 21 érték a `myL` lista elemei között. Adjuk meg a típusdefiníciókat is.

```
fo = fugv myL 21 0 ...
   where myL = [1,5,7,9,10,11,21,21,21,21,34,55,67,90]

fugv ls e x y
    | y < x      = ...
    | k > e      = fugv ls e x (j-1)
    | k < e      = fugv ls e (j+1) y
    | otherwise  = j
    where
    j = x + ((y - x) `div` 2)
    k = ...
```

6. Írjunk egy függvényt, amely a bemeneti számpár típusú listát kiírja a képernyőre, úgy hogy minden számpárt külön sorba ír, az elemek közé vesszőt és space-et téve. Használjuk a következő típusdefiníciót:

```
myPutStr :: (Show a, Show b) => [(a, b)] -> IO()

Példa:
> myPutStr [(1.5,2), (3.3,4), (9.3, 10), (12.5, 21)]
1.5, 2
3.3, 4
9.3, 10
12.5, 21
```

7. Alkalmazva a következő felhasználó által definiált típust, írjunk függvényt, amely meghatározza az adott márkájú telefonok számát:

```
data Telef = Telef{
    nev :: [Char],
    ar :: Int,
    kod :: Int
    }deriving (Show)
```

A függvény szignatúrája legyen a következő:

```
szamol :: [Char] -> [Telef] -> Int
```