Задачи за подготовка за второ контролно по ФП,
специалност „Информационни системи“

Задача 1. Нека f е функция от тип Int **->** Int, а lst е списък от цели числа $[a_1,a_2, \dots ,a_n]$. Дефинирайте функция **func f lst**, която връща функция от тип Int **->** Int, чиято стойност за всяко цяло число x е равна на $f(a_1x) + 2.f(a_2x) + \dots + n.f(a_nx)$.

Задача 2. Нека f е функция от тип Int **->** Int, а lst е списък от цели числа $[a_1,a_2, \dots ,a_n]$. Дефинирайте функция **func f lst**, която връща функция от тип Int **->** Int, чиято стойност за всяко цяло число x е равна на $a_1f(x) + a_2f(x^2) + \dots + a_nf(x^n)$.

Задача 3. Дефинирайте функция от по-висок ред **boundUp f y**, която приема едноаргументна числова функция f и число y и връща като резултат едноаргументна числова функция, чиято стойност във всяка дадена точка x е равна на стойността на f в x, ако тази стойност е по-голяма от y, или на y в противен случай.

Задача 4. Разглеждаме непразен списък от едноаргументни функции $[f_1, f_2, \dots , f_n]$, всяка от които е от тип Int -> Int. Дефинирайте функция **getOddCompositionValue**, която при подаден такъв списък lst връща като резултат функция, чиято стойност за всяко цяло число x е равна на стойността на композицията на функциите с нечетни поредни номера от lst в x, т.е. на $f_1(f_3(\dots (x) \dots))$.
Пример:
getOddCompositionValue [(\x -> x+1),(\x -> x*2),(\x -> x-1)(\x -> x ‘div’ 2)] 2 → 2

Задача 5. Нека са дефинирани типовете:
```
data Color = Red | Green | Blue                    -- цвят
  deriving (Read, Show, Eq)
data Tree = Empty | Node Color Tree Tree           -- двоично дърво от тип Color
```
Дефинирайте функция **minDepthGreenNode** :: Tree **->** Int, която намира дълбочината на най-плиткия (най-близкия до корена) връх с цвят Green на дадено двоично дърво от тип Color.

Задача 6. Нека са дефинирани типовете:
```
data Color = Red | Green | Blue                    -- цвят
  deriving (Read, Show, Eq)
data Tree = Empty | Node Color Tree Tree           -- двоично дърво от тип Color
```
Дефинирайте функция **maxDepthBlueNode** :: Tree **->** Int, която намира дълбочината на най-дълбокия (най-отдалечения от корена) връх с цвят Blue на дадено двоично дърво от тип Color.

Задача 7. Нека са дефинирани типовете:
```
type Name = String                                 -- име
type Capital = Name                                -- столица
type AvgYearlyTemperature = Double                 -- средногодишна температура
type Elevation = Int                               -- надморска височина
data City = City Name Elevation AvgYearlyTemperature   -- град
  deriving (Read, Show)
data Country= Country Name Capital [City]          -- държава
  deriving (Read, Show)
```
Дефинирайте функция **coldestCapital** :: [Country] **->** Name, която получава като аргумент списък от държави и връща като резултат името на държавата от списъка с най-студена столица (столица с най-ниска средногодишна температура).

Задача 8. Нека са дефинирани типовете:

```
type Name = String                                          -- име
type Capital = Name                                         -- столица
type AvgYearlyTemperature = Double                          -- средногодишна температура
type Elevation = Int                                        -- надморска височина
data City = City Name Elevation AvgYearlyTemperature        -- град
  deriving (Read, Show)
data Country= Country Name Capital [City]                   -- държава
  deriving (Read, Show)
```

Дефинирайте функция highestCapital :: [Country] **->** Name, която получава като аргумент списък от държави и връща като резултат името на държавата от списъка с най-висока столица (столица с най-голяма надморска височина).