## Робота №01

*Списки*

Розглядаються списки натуральних чисел:

* Береться будь-яке натуральне число n – це перший елемент списку.
* Якщо воно парне, то наступний елемент - число n `div` 2, в іншому випадку - число n\*3+1.
* Процес формування списку чисел закінчується, коли отримано число 1.
* Для n==3 маємо список [3, 10, 5, 16, 8, 4, 2, 1]

Такий список натуральних чисел називається *числа-градини*, або Сіракузька послідовність. Будемо говорити, що цей список числа-градини породжує його перший елемент. Наприклад, список числа-градини [3, 10, 5, 16, 8, 4, 2, 1] породжує число 3.

На основі допоміжного файлу створити файл, в якому надати визначення наступних функцій.

1. Функція *power3,* котра будує нескінченний список третьої степені натуральних чисел. Наприклад
   * take 4 power3 = [1, 8, 27, 64]
2. Функція *toPower3,* котра будує нескінченний список натуральних степенів числа три*.* Наприклад:
   * take 4 toPower3 = [3, 9, 27, 81]
3. Функція *sumPower3 n,* котра обраховує суму ряду . Можна скористатися функцією *sum xs*, що рахує суму всіх елементів списку *xs*. Наприклад:
   * sumPower3 4 = 120
4. Функція *sumPower m n,* котра обраховує суму ряду , де *F(m,i) = mi.* Передумова - *m≥0.* Можна скористатися функцією *sum.* Наприклад:
   * sumPower 2 5 = 62
5. Функція *lessMe xs*, котра знаходить для кожного елементу списку *xs*, скільки є елементів строго менших за нього, і виводить в результуючий список на відповідну позицію цю кількість. Наприклад:
   * lessMe [1,5,3,4,3] = [0,4,1,3,1]
6. Функція *frequency xs*, що за списком цілих чисел *xs,* повертає список пар (елемент, частота). Кожна пара визначає елемент із списку *xs* і кількість (частоту) його входжень в список *xs*. Можна скористатися функцією *length.* Наприклад:
   * frequency [1,1,2,2,1,1] = [(1,4),(2,2)]
7. Функція *hailstone n,* що формує наступне число, за числом *n* в списку числа-градини. Наприклад:
   * hailstone 3 = 10
8. Функція *hailSeq n*, що будує список числа-градини, котрий породжує число *n* (цей список починається з числа *n*). Можна скористатися функцією *hailstone*. Наприклад:
   * hailSeq 3 = [3, 10, 5, 16, 8, 4, 2, 1]
9. Функція *allHailSeq*, котра будує нескінченний список всіх списків числа-градини , що породжуються натуральними числами 1,2, … Можна скористатися функцією *hailSeq*. Наприклад:
   * take 5 allHailSeq = [[1], [2,1], [3,10,5,16,8,4,2,1], [4,2,1], [5,16,8,4,2,1]]
10. Функція *firstHailSeq l*, котра знаходить мінімальне число, що породжує список числа-градини довжини *l.* Можна скористатися функцією *allHailSeq.* Наприклад:
    * firstHailSeq 12 = 48

просто біжимо по масиву, якщо

*power3, toPower3*:: [Integer]

*sumPower3*  :: Integer -> Integer

*sumPower*  :: Integer -> Integer -> Integer

*lessMe*  :: [Int] -> [Int]

*frequency*  :: [Int] -> [(Int,Int)]

*hailstone, firstHailSeq* :: Int -> Int

*hailSeq*  :: Int -> [Int]

*allHailSeq* :: [[Int]]

Зауваження:

Назва файлу Family01.hs (Family – прізвище студента). Файл включає модуль Family01 і створюється на основі файлу-заготовки HWP01.hs

-- Задача 4 -----------------------------------------

sumPower :: Integer -> Integer -> Integer

sumPower = undefined

-- Задача 5 -----------------------------------------

lessMe :: [Int] -> [Int]

lessMe = undefined

-- Задача 6 -----------------------------------------

frequency :: [Int] -> [(Int,Int)]

frequency = undefined

-- Задача 7 -----------------------------------------

hailstone :: Int -> Int

hailstone = undefined

-- Задача 8 -----------------------------------------

hailSeq :: Int -> [Int]

hailSeq = undefined

-- Задача 9 -----------------------------------------

allHailSeq :: [[Int]]

allHailSeq = undefined

-- Задача 10 -----------------------------------------

firstHailSeq :: Int -> Int

firstHailSeq = undefined