Контролна работа № 1 по Функционално програмиране
Специалност „Информационни системи“, I курс, 06.04.2024 г.

Задача 1

Алгоритъм за “компресиране” на число работи по следния начин:

1. Ако числото е четно, го дели на 2.
2. Ако числото е нечетно, го умножава по 3 и добавя 1.

Стъпките се повтарят, докато числото не стане 1. Например, ако подаденото число е 3, то числата, през които се преминава до достигане на 1, са: 10, 5, 16, 8, 4 и 2. “Странно число” е всяко число, за което броят на числата в редицата, необходими за достигане до 1, е четен. “Красиво число” е всяко число, за което броят на числата в редицата, необходими за достигане до 1, е нечетен. “Оригинално число” е всяко красиво число, по-голямо от 1, за което е вярно, че двете естествени числа непосредствено преди и след него са странни. Например 5 е оригинално число, защото е красиво и числата 4 и 6 са странни.

Да се дефинира функция **`sumOriginalNumbers :: Int -> Int -> Int`**, която приема две цели числа, по-големи от 1 – `start` и `finish`, и намира сумата на оригиналните числа в интервала [`start`, `finish`]. Функцията да реализира линеен итеративен процес. При подаден невалиден вход да се поражда грешка с подходящ текст.

Примери:

```
sumOriginalNumbers 3 100     → 586
sumOriginalNumbers 25 250    → 1497
sumOriginalNumbers 1000 2000 → 93585
sumOriginalNumbers 15 2222   → 152464
```

Задача 2

Да се дефинира функция **`getDistribution :: Int -> ((Int -> Int) -> [(Int, Int)])`**, която приема неотрицателно цяло число `n` и връща нова функция, която приема унарна функция `f` и връща списък с елементи от вида (<цифра> . <брой срещания в `f(n)`>), представящ разпределението на цифрите на `f(n)`.

Примери:

```
(getDistribution 123) (\ x -> x - 5) → [(1,2),(8,1)]
(getDistribution 232) (*3) → [(6,2),(9,1)]
(getDistribution 0) (2^) → [(1,1)]
(getDistribution 881122) (\ x -> x - 10014) →
[(0,1),(1,2),(7,1),(8,2)]
(getDistribution 881122) (10 `mod`) → [(0,1),(1,1)]
```

Задача 3

Даден е краен списък от цели числа и жаба на първата позиция:

```
[цяло число, цяло число, ... , цяло число]
```

Всяко число в списъка означава дължината и посоката на скок, който жабата прави. Например, ако стойността е:

- `2`, жабата прави скок с две позиции надясно;
- `-3`, жабата прави скок с три позиции наляво;
- `0`, жабата остава на същата позиция.

Да се дефинира функция **`frogJumps :: [Int] -> Int`**, която намира броя скокове, необходими за излизане от списъка. Ако жабата не може да изскочи от списъка, да се връща -1.

Примери:

```
frogJumps [1, 2, 1, 5] → 3 -- скокове = 3 (1 -> 2 -> 5 -> <излизане
от списъка>)
frogJumps [1, 2, 2, -1] → 4
frogJumps [1, -1] → -1
frogJumps [2, -3, -1, 4, 5, 2, 3, 6, 3, 2, -2, 3, -1, -2, 8, 3, 4,
4, 3, 9, 3] → 3
frogJumps [0, 9, 81, -82, 38, -50, -27, 29, -27, -88, -72, 54, -97]
→ -1
frogJumps [2, 60, -2, -39, 78, -19, 99, -32, 48, -85, 50, 57] → -1
frogJumps [30, 5, 17, 80, -4, 50, -15, 23, 84, -49, 3, 71, 97, -3,
-24, 45, -38, -46, 19, 98, 65, 7, -31, -59, -51, -80, 42, -76, -90,
-14, 0, 84, -27, -90, 36] → -1
frogJumps [3, -69, 22, 32, -3, -50, 51, 75, -82, -67, -77, 10, 16,
-72, -65, 2, -67, 2, -28, 37, 76, -72, -44, 51, -53, -8, -60, 74,
-53, 95, 40, 97, 63, -56, 34, -32, 80, 46, -17, -95, 6] → -1
```

Задача 4

Система за оценка на броя точки, получени от набор скречкарти, работи по следния начин:

1. Приема се на вход списък от скречкарти, всяка от които се състои от двойка (вектор) от два списъка: първият списък представя печелившите числа, а вторият – изтеглените числа.
2. За всяка карта се изчисляват спечелени точки. Те се изчисляват чрез изтеглените числа, които присъстват и като печеливши. Първото число, което е изтеглено и се среща като печелившо, дава една спечелена точка. Всяко следващо съвпадение удвоява спечелените точки.
3. След като всички карти са оценени, системата връща общата сума на спечелените точки.

Да се дефинира функция **`totalPoints :: [([Int],[Int])] -> Int`**, която приема списък от скречкарти и намира общата сума на спечелените от тях точки.

Примери:

```
totalPoints [
  ([41,48,83,86,17],
   [83,86,6,31,17,9,48,53]),
  ([13,32,20,16,61],
   [61,30,68,82,17,32,24,19]),
  ([1,21,53,59,44],
   [69,82,63,72,16,21,14,1]),
  ([41,92,73,84,69],
   [59,84,76,51,58,5,54,83]),
  ([87,83,26,28,32],
   [88,30,70,12,93,22,82,36]),
  ([31,18,13,56,72],
   [74,77,10,23,35,67,36,11])] → 13

totalPoints [
  ([11,54,66,33,51,59,82,24,3,88],
   [70,28,24,89,66,42,22,59,88,33,99,54,31,11,39,3,51,82,38,16,68]),
  ([26,7,38,74,20,89,78,79,73,47],
   [88,64,13,18,99,9,37,61,60,97,22,67,48,
       95,19,76,40,31,6,90,42,2,41,1,68]),
  ([67,65,8,4,84,62,69,66,46,36],
   [27,30,2,16,45,99,65,50,37,19,78,87,49,
       64,12,84,11,8,4,69,44,62,48,71,17])] → 544

totalPoints [
  ([92,89,2,29,25,53,65,30,38,71],
   [11,53,6,63,15,50,41,37,27,96,73,57,64,85,
       59,1,22,49,25,52,29,80,72,58,28]),
  ([36,66,57,82,10,1,28,25,56,83],
   [23,58,38,35,97,66,55,14,85,79,54,77,93,62,67,4,
       11,99,94,90,32,22,12,36,63]),
  ([61,19,76,17,81,18,87,44,45,74],
   [37,15,31,67,24,4,77,81,63,68,27,94,3,62,12,90,
       69,2,8,34,60,53,97,43,73]),
  ([61,97,73,13,88,93,19,75,47,89],
   [68,79,15,25,59,16,78,5,40,69,92,20,4,58,22,30,
       67,21,76,44,81,98,65,74,46]),
  ([3,45,59,47,12,65,8,57,98,53],
   [77,96,12,8,22,63,59,88,61,43,66,39,90,45,55,
       47,10,73,53,34,25,3,6,99,38])] → 71
```

Обяснение за първия пример:

Карта `1` има пет печеливши числа (`41`, `48`, `83`, `86` и `17`) и осем числа, които са изтеглени (`83`, `86`, `6`, `31`, `17`, `9`, `48` и `53`). Четири от изтеглените числа – `48`, `83`, `17` и `86`, са и печеливши числа. Това означава, че карта `1` носи `8` точки (`1` точка от първото съвпадение, която след това се удвоява трикратно за всяко от трите съвпадения след първото). Карта `2` има две печеливши числа (`32` и `61`) и носи `2` точки. Карта `3` има две печеливши числа (`1` и `21`) и носи `2` точки. Карта `4` има едно печеливше число (`84`) и носи `1` точка. Карти `5` и `6` не съдържат печеливши числа и не носят точки. Общата сума на спечелените точки от картите е `13` точки.