Изпит по Функционално програмиране
Специалност „Информационни системи“, I курс, 24.06.2021 г.

Задача 1. Да се дефинира функция `applyEveryKth :: (a -> a) -> Int -> [a] -> [a]`, която получава едноаргументна функция `f`, естествено число `k` и списък `xs`. Дефинираната функция трябва да върне нов списък, състоящ се от елементите на `xs`, но в който към всеки `k`-ти елемент на `xs` е приложена функцията `f`.

Примери:

```
applyEveryKth (* 2) 3 [1..6] ⟶ [1,2,6,4,5,12]

applyEveryKth (+ 2) 4 [1..7] ⟶ [1,2,3,6,5,6,7]
```

Задача 2. Да се дефинира функция `speak :: String -> (Char -> String)`, която получава символен низ `str` и връща нова **анонимна функция**, която приема символ `c` и заменя всяко негово срещане в `str` със съответния брой на символите, оставащи до края на `str`. Задачата да се реши чрез **foldl**!

Примери:

```
(speak "gate") 't'            ⟶ "ga1e"

(speak "This is a test") 'i' ⟶ "Th11s 8s a test"

(speak "iiiiiii") 'i'         ⟶ "6543210"

(speak "This is another test that has more words") 'a' ⟶ "This is
31nother test th16t h12s more words"
```

Задача 3. Имате съквартирант, който много обича да готви. Любимите му ястия са ябълков пай, бургер и пиле. Тези ястия толкова много му харесват, че той готви само тях в следната последователност: ябълков пай, бургер, пиле, отново ябълков пай и т.н. Ето как той преценява точно кое ястие да сготви:

- ако времето е слънчево, вместо да прави друго ястие, той предпочита да сготви това, което е готвил предишния ден, и да излезе навън с приятели;
- ако времето е дъждовно, той предпочита да стои вкъщи и да сготви следващото ястие.

Да се дефинира функция `cook` , която приема списък от ястия и връща списък, който представя данни за времето през съответните дни, щом са били сготвени подадените ястия. Може да приемете, че подаденият списък винаги ще съдържа поне две ястия.

В коментар опишете какви алгебрични типове използвате, за да решите задачата.

Примери:

```
cook [ApplePie, ApplePie, Burger, Chicken, Chicken, ApplePie] ⟶
[Sunny, Rainy, Rainy, Sunny, Rainy]

cook [ApplePie, Burger, Chicken, Chicken, ApplePie, Burger] ⟶
[Rainy,Rainy,Sunny,Rainy,Rainy]
```

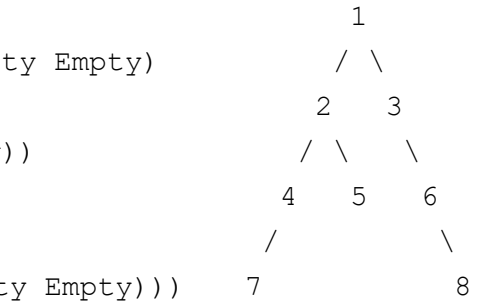
Задача 4. Нека е дадено следното представяне на двоично дърво:

```
data BTree = Empty | Node Int BTree BTree.
```

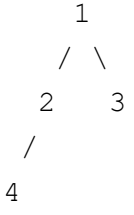
Да се дефинира функция `deepestNodesSum :: (Int -> Bool) -> BTree -> Int`, която получава предикат `p` и двоично дърво `bt` и връща сумата на стойностите в най-отдалечените от корена възли на `bt`, за които предикатът `p` връща „истина“.

Примери:

```
t1 :: BTree
t1 = Node 1 (Node 2 (Node 4 (Node 7 Empty Empty)
                            Empty)
                    (Node 5 Empty Empty))
            (Node 3 Empty
                    (Node 6 Empty
                            (Node 8 Empty Empty)))
```




```
t2 :: BTree
t2 = Node 1 (Node 2 (Node 4 Empty Empty)
                    Empty)
            (Node 3 Empty Empty)
```




```
deepestNodesSum odd t1  ⟶ 7
deepestNodesSum even t2 ⟶ 4
```