Контролна работа № 2 по Функционално програмиране
Специалност „Информационни системи“, I курс, 03.06.2023 г.

Задача 1.

Да се дефинира функция **`prodOdds :: Num a => [a] -> a`**, която приема списък от числа и намира произведението на числата, намиращи се на позиции с нечетен индекс в списъка. Списъкът е индексиран от `0`. Функцията да се дефинира **на функционално ниво** и в решението да се използва **foldr**. За получаване на пълен брой точки, проверката за нечетност трябва да се извърши в първия аргумент на **foldr**!

Примери:

```
prodOdds [1,2,3,4,5,6]    → 48
prodOdds [7.66,7,7.99,7] → 49.0
```

Задача 2.

Разглеждаме думата *abode*. В нея буквата *a* е на позиция *1*, а *b* е на позиция *2*. В английската азбуката *a* и *b* също са на позиции съответно *1* и *2*. Забелязваме също, че и *d*, и *e* в *abode* заемат позициите, които биха заели в азбуката: съответно *4* и *5*.
Да се дефинира функция **`solve :: [String] -> [Int]`** , която приема списък от думи и връща списък с броя букви, които заемат своите позиции в азбуката за всяка дума. Входните данни ще бъдат съставени само от думи, включващи главни и малки букви от английската азбука.

Примери:

```
solve ["abode","ABc","xyzD"]                          → [4,3,1]
solve ["abide","ABc","xyz"]                           → [4,3,0]
solve ["IAMDEFANDJKL","thedefgh","xyzDEFghijabc"] → [6,5,7]
solve ["encode","abc","xyzD","ABmD"]                  → [1,3,1,3]
```

Задача 3.

Дефиниран е полиморфен алгебричен тип **`NTree a`**, описващ дърво с произволен брой наследници:

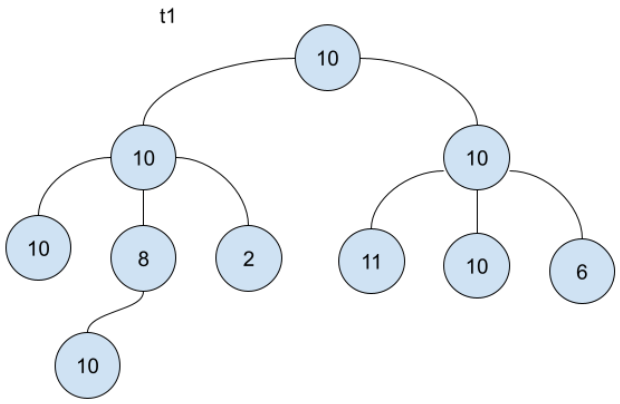
```
data NTree a = Nil | Node a [NTree a] deriving (Eq, Show).
```

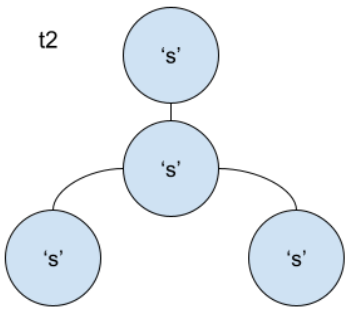
Да се дефинира функция **`mapLevel :: NTree a -> [(a -> b)] -> NTree b`**, която приема дърво с произволен брой наследници и списък от унарни функции и връща ново дърво, за което е вярно, че за всички възли на височина `i` е приложена `i`тата функция от списъка (т.е. за възлите на височина `1` е приложена първата функция, за възлите на височина `2` - втората и т.н.). Коренът се намира на височина `1`. В случай, че няма

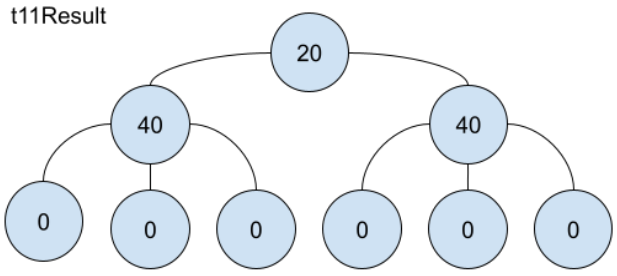
съответна функция или възли на дадена височина, да се връща празно дърво. Ако даден възел е лист (няма разклонения след него), то неговият списък с наследници трябва да е празният списък (в който не се съдържа Nil).

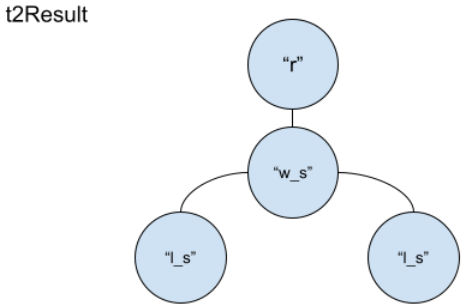
Примери:

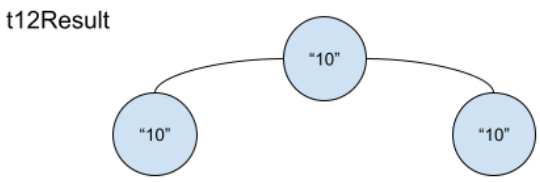
```
t1 = Node 10 [Node 10 [Node 10 [], Node 8 [Node 10 []], Node 2 []],
Node 10 [Node 11 [], Node 10 [], Node 6 []]]
t2 = Node 's' [Node 's' [Node 's' [], Node 's' []]]

t11Result = Node 20 [Node 40 [Node 0 [],Node 0 [],Node 0 []],Node 40
[Node 0 [],Node 0 [],Node 0 []]]
t12Result = Node "10" [Node "10" [],Node "10" []]
t2Result = Node "r" [Node "w_s" [Node "l_s" [],Node "l_s" []]]

mapLevel t1 [(*2), (*4), (`div` 100)] → t11Result
mapLevel t1 [show, (nub . show . (* 1000))] → t12Result
mapLevel t2 [(\ _ -> "r"), (\ char -> "w_" ++ [char]), (\ char ->
"l_" ++ [char])] → t2Result
```











Задача 4.

Наети сте от фирма, произвеждаща електрически гаражни врати. Инцидентите с настоящата продуктова линия са довели до множество повредени коли, навехнати крайници и няколко уплашени домашни любимци. Задачата Ви е да напишете по-безопасна версия на софтуера, контролиращ вратите.

Характеристиката на процеса е следната:
- Дистанционното управление има точно един бутон - `P`.
- Винаги се започва със затворена врата.
- Ако вратата е затворена, натискането започва да я отваря, а ако е отворена - да я затваря.
- Отнема 5 секунди, за да се отвори или затвори вратата напълно.
- Докато вратата се движи, едно натискане спира движението, а последващо възобновява движението в същата посока.
- Вратата има сензор, с който открива препятствия. Когато вратата открие препятствие, тя трябва незабавно да обърне посоката на движение.
- Вратата започва да се движи веднага, следователно нейната позиция се променя в същата секунда, в която се случва събитието.

Да се дефинира функция **`controller :: String -> String`**, която приема низ, в който всеки знак представя една секунда, със следните възможни стойности.
- `'.'`: няма събитие;
- `'P'`: бутонът `P` е натиснат;
- `'O'`: открито е препятствие.

Например, `'..P....'` означава, че нищо не се случва в продължение на две секунди, след това бутонът е натиснат и няма други събития.

Функцията трябва да върне низ, в който всеки знак показва позицията на вратата за всяка секунда. Възможните позиции варират от `0` (напълно затворена) до `5` (напълно отворена).

Примери:

```
controller ""                      →  ""
controller ".........."            → "0000000000"
controller "P...."                 → "12345"
controller "P.P.."                 → "12222"
controller "..P...O..."            → "0012343210"
controller "P......P......"        → "123455543210000"
controller "P.P.P...."             → "122234555"
controller ".....P.P........P...." → "00000122222222234555"
controller ".........."            → "0000000000"
controller "P.."                   → "123"
controller "P...."                 → "12345"
controller "P......P......"        → "123455543210000"
```

```
controller "P.P.."                  → "12222"
controller "P.P.P...."              → "122234555"
controller ".....P.P........P...."  → "000001222222222234555"
controller ".....P......P.P..P...." → "000001234555433321000"
controller "P.O...."                → "1210000"
controller "P......P.O...."         → "12345554345555"
controller "P..OP..P.."             → "1232222100"
controller "P......P..OP..P..."     → "1234555432333334555"
controller "..P...O....."           → "001234321000"
```