Започнат на	понеделник, 16 август 2021, 08:02
Състояние	Завършен
Приключен на	понеделник, 16 август 2021, 08:45
Изминало време	42 мин. 32 сек.

Въпрос **1**

Отговорен

От максимално 1,00

Реализирайте описаните по-долу функции на посочения за всяка от тях език.

Изисква се да представите решения и на трите задачи!

Задача 1 *(Scheme)*: Нека са дадени две естествени числа M и N. Ще казваме, че M "съдържа" N, ако цифрите на десетичния запис на N се срещат някъде в десетичния запис на M, точно в същия ред и непосредствено една до друга. Например, ако N=123, то 123, 098212309, 1230 и 56123 съдържат N. От друга страна 15253, 12, 321 и 1243 НЕ съдържат N.

Реализирайте функция `(pattern-stream n)`, която получава естествено число n > 0 и връща поток от всички естествени числа, които съдържат n, в смисъла на горната дефиниция. Потокът да съдържа числата в строго нарастващ ред. Задачата да се реши на език `racket/base`. За генерирането на потока използвайте наготово функциите от библиотеката `racket/stream` (`stream-cons`, `stream-take`, `stream->list`, `stream-first` и т.н.).

Упътване: напишете предикат, който проверява дали дадено естествено число съдържа n в записа си.

Задача 2 *(Haskell)*: Довършете кода на функцията `quickSort` така, че тя да сортира списък по метода на бързото сортиране (quicksort). Опишете типа на функцията по такъв начин, че тя да може да работи със списък от произволен тип елементи, стига те да са от класа `Ord`. В решението не може да се добавя друг код, освен на подчертаните места, вкл. не може да се дефинират други функции.

Упътване: Използвайте x в качеството на делителен (pivot) елемент.

```
quickSort :: ____________________________________________
quickSort []     = ____________________________________________
quickSort (x:xs) = _________________ ++ _____________++ _________________
    where
        lesser  = filter ____________________ ____________________
        greater = filter ____________________ ____________________
```

Задача 3 *(Haskell)*: Двоично дърво ще представяме чрез следния тип:

```
data Tree a = EmptyTree | Node {
                            value :: a,
                            left  :: Tree a,
                            right :: Tree a
                          } deriving (Show,Read)
```

Довършете функцията `treeWords`. Опишете типа на функцията така, че тя да получава дърво от символи и да връща списък от символни низове. Функцията да връща списък от всички думи, които могат да се образуват по път от корена, до някое от листата на дървото. Ако дървото е празно, да се връща празният списък. В решението не може да се добавя друг код, освен на подчертаните места, вкл. не може да се дефинират други функции.

```
treeWords :: ____________________________________________________
treeWords ____________________________________ = []
treeWords (Node v EmptyTree EmptyTree) = [[_____]]
treeWords (Node v l r)  = map (v:) (_________________ ++ _________________)
    where wl = ____________________________________
          wr = ____________________________________
```

45557.rkt

45557.hs

◄ График за първа част на изпита (16.08.2021 г.)

Отиди на ...

График за теоретичния изпит ►