Домашна работа № 3 по Функционално програмиране

специалност „Информационни системи“, I курс, 2022/2023 учебна година

Решенията трябва да са готови за автоматично тестване. Важно е програмният код да бъде добре форматиран и да съдържа коментари на ключовите места. Предайте решенията на всички задачи в *един* файл с наименование *hw3_<FN>.hs*, където *<FN>* е Вашият факултетен номер.

Домашните работи се предават като изпълнение на съответното задание в курса по ФП в Moodle (https://learn.fmi.uni-sofia.bg/course/view.php?id=9029) най-късно до 23:55 ч. на 06.06.2023 г. (вторник).

Приятна работа и успех!

Задача 1

Крайно кореново дърво, всеки връх на което съдържа цяло число и може да има произволен брой деца, се представя със следната рекурсивна полиморфна структура:

```
data NTree a = T a [(NTree a)]
```

Лист наричаме дърво с единствен връх, а **пръчка** наричаме дърво, в което всеки връх има най-много едно дете. Казваме, че едно дърво се **окастря**, ако всички пръчки в него, които са с повече от два върха и не са част от други пръчки, се скъсят откъм листата до дължина точно два върха. Дефинирайте функция **`prune :: NTree a -> NTree a`**, която връща окастрено копие на дървото, подадено като параметър.

Пример:

```
prune T 1 [T 2 [T 3 []], T 4 [T 5 [T 6 []]], T 7 [T 8 [], T 9 [T
10 [T 11 []]]]] → T 1 [T 2 [T 3 []], T 4 [T 5 []], T 7 [T 8 [], T
9 [T 10 []]]]
```

Задача 2

Нека е даден низ, който представя аритметичен израз. Аритметичният израз се състои само от:

- едноцифрени цели числа (символите от `0` до `9`);
- операциите за събиране (символ `+`) и изваждане (символ `-`);
- отваряща и затваряща скоба (`(` и `)`), които задават приоритет на операциите;
- символни променливи (символ от `a` до `z`).

Дефинирайте функция `simplify :: String -> String`, която опростява подадения аритметичен израз `inp`, до израз `out`, за който е изпълнено следното:

- стойността на `inp` е еквивалентна на `out`;
- не съдържа скоби;
- съдържа операциите за събиране и изваждане;
- всяка от символните променливи се среща само по веднъж, като се предхожда от съответния множител (освен когато е `1` или `-1`), така че стойностите на `inp` и `out` да са равни;
- символните променливи в out са подредени по азбучен ред;
- съдържа най-много едно цяло число, което не е последвано от символна променлива, и то е винаги накрая на израза.

Примери:

```
simplify "1+2+x" → "x+3"
simplify "x+2+x-2" → "2x"
simplify "x+2-(x-2)" → "4"
simplify "y+2+x-2" → "x+y"
simplify "1+2+x+y+x+z+5-x-x-x+y" → "-x+2y+z+8"
simplify "1+2+x+y+x-(x-x-x)+z+y-9" → "3x+2y+z-6"
simplify "1+2-(3-(3-2))-9" → "-8"
```