Programowanie funkcyjne — kolokwium nr 1, 9.12.2025

Instrukcja: Każde zadanie należy przesłać na Pegaza w oddzielnym pliku: zadanie1.hs, zadanie2.hs i zadanie3.hs. Plików nie należy zipować. Rozwiązania muszą się poprawnie kompilować. W rozwiązaniach nie można korzystać z modułów innych niż standardowe; niedozwolone jest użycie polecenia import. Rozwiązania nie spełniające powyższych wymogów nie będą oceniane. Użycie polskich znaków w sygnaturach nie jest obowiązkowe. Czas na rozwiązanie zadań: 75 minut. Punktacja: 10 punktów za każde zadanie. Uwaga: korzystanie z internetu (poza wyznaczonym czasem) lub z AI skutkuje automatycznym otrzymaniem 0 punktów.

Zadanie 1. Napisać funkcję o sygnaturze

`różnica :: String -> Integer`,

która dla znakowej reprezentacji liczby naturalnej zwraca różnicę sumy jej cyfr na pozycjach nieparzystych oraz sumy jej cyfr na pozycjach parzystych (licząc pozycje od lewej, skrajna lewa to pierwsza). Przykładowo `różnica "123"` ma zwrócić $2 = 1 + 3 - 2$, a `różnica "1234"` ma zwrócić $-2 = 1+3-(2+4)$. Używając tej funkcji oraz funkcji `show` napisać funkcję o sygnaturze

`sprawdź :: [Integer] -> [String]`,

zamieniającą listę liczb nieujemnych na listę napisów: `"zero"`, jeśli funkcja `różnica` na ich reprezentacji znakowej zwraca 0, oraz `"nz"` w przeciwnym przypadku. Przykładowo, `sprawdź [123,1232]` ma zwrócić `["nz","zero"]`.

Zadanie 2. Napisać funkcję o sygnaturze

`podciąg :: (a -> Bool) -> [a] -> [a]`,

która zwraca najdłuższy spójny podciąg listy taki, że wszystkie jego elementy spełniają podany predykat (jeśli jest więcej niż jeden taki podciąg, to można zwrócić dowolny). Przykładowo, wywołanie `podciąg (>0) [1,0,2,3,4,-3,1,2]` powinno zwrócić `[2,3,4]`.

Zadanie 3. *Cepem dwuliniowym* nazywamy strukturę danych, która gromadzi elementy podobnie do listy, ale automatycznie rozdziela je na dobre i złe według podanego przy inicjalizacji predykatu rozdzielającego. Cep pozwala odczytać listę elementów dobrych oraz listę elementów złych, a także zapytać o status ostatnio dołożonego elementu i ewentualnie zmienić go na element dobry. Cep implementuje nastepujące operacje: `cp` zwraca cep pusty z wbudowanym predykatem; `de` dokłada element; `cd` mówi, czy ostatnio dołożony element jest dobry; `zd` ustawia status ostatnio dołożonego elementu jako dobry (niezależnie od jego dotychczasowego statusu); `ld` i `lz` zwracają listę elementów dobrych/złych, zgodnie z kolejnością dokładania. Z wyjątkiem `ld` i `lz`, wszystkie operacje działają w czasie stałym, z tym że pomijamy czas obliczania predykatu rozdzielającego.

Zdefiniować typ `Cep a`, służący do przechowywania elementów typu `a` w cepie dwuliniowym, oraz następujące funkcje, realizujące opisane wyżej operacje z odpowiednią złożonością:

```
cp :: (a -> Bool) -> Cep a
de :: a -> Cep a -> Cep a
cd :: Cep a -> Bool
zd :: Cep a -> Cep a
ld :: Cep a -> [a]
lz :: Cep a -> [a]
```

Przykładowo, wywołanie `ld $ de 1 $ de 2 $ cp even` powinno zwrócić listę `[2]`, zaś wywołanie `ld $ zd $ de 1 $ de 2 $ cp even` — listę `[1,2]`.