Programowanie funkcyjne — kolokwium nr 2, 24.01.2018

Instrukcja: Rozwiązania zadań należy przesłać do godziny 11:00 na adres `kolokwium.pf@gmail.com` (decyduje data stempla googlowego). Całość należy przesłać w jednym mailu, ale każde zadanie w oddzielnym pliku: zad1.hs, zad2.hs i zad3.erl. Plików nie należy zipować. W rozwiązaniach nie można korzystać z modułów innych niż standardowe; niedozwolone jest użycie polecenia `import`. Zadania 1 i 2 napisać w Haskellu, zadanie 3 — w Erlangu. Rozwiązania nie spełniające powyższych wymogów nie będą oceniane. Punktacja: 10 punktów za każde zadanie. Uwaga: korzystanie z internetu poza wyznaczonym czasem skutkuje automatycznym otrzymaniem 0 punktów.

Zadanie 1. Funkcja prefixes zwraca listę list zawierającą wszystkie prefiksy (początkowe fragmenty) podanej listy ℓ, od listy pustej aż do ℓ. Przykładowo:

```
prefixes [1,2,3,4] = [[], [1], [1,2], [1,2,3], [1,2,3,4]]
prefixes "tekst" = ["", "t", "te", "tek", "teks", "tekst"]
```

Napisać funkcję prefixes bezpunktowo. Kolejność prefiksów w wynikowej liście nie ma znaczenia.

Wskazówka: wykorzystać fold oraz map.

Zadanie 2. Funkcja compr dla podanej listy akcji w monadzie m, tzn. funkcji typu $a \to m\ a$, zwraca akcję, która jest „złożeniem" (od prawej do lewej) wszystkich akcji podanych na liście. Dokładniej, dla listy akcji $[a_1, a_2, ..., a_n]$ oraz argumentu x, wynikowa akcja compr $[a_1, a_2, ..., a_n]$ na argumencie x powinna działać następująco:

- aplikuje akcję a_n do x, otrzymując wartość x_{n-1}, opakowaną przez m
- aplikuje akcję a_{n-1} do x_{n-1}, otrzymując x_{n-2}, opakowane przez m
- ... i tak dalej aż do końca listy, gdzie aplikuje a_1 do x_1, otrzymując ostateczny wynik.

Przykładowo:

```
compr [\x->Just (x+1), \x->Just (2*x)] 1 = Just 3
compr [\h->putStrLn ("Dlugosc: " ++ show(length h)) >> return h,
       \t->putStr t >> getLine] "Podaj tekst: "
```

(drugie wywołanie spowoduje wypisanie pytania, oczekiwanie na wprowadzenie tekstu i wyświetli informację o jego długości; wynikiem działania funkcji jest wprowadzony tekst „opakowany" przez IO). Jeśli lista akcji jest pusta, funkcja powinna zwrócić jako wynik akcję return. Napisać funkcję compr oraz określić jej najogólniejszą możliwą sygnaturę.

Zadanie 3. Napisać moduł, który uruchomi N procesów ($N \geqslant 3$). Każdy z procesów ma odbierać listę liczb i wypisywać ją na konsolę, a następnie dołączać na początku listy sumę dotychczasowych elementów i przekazywać listę nastepnemu procesowi w cyklu (w nieskończonej pętli). Moduł ma udostepniać funkcję start(N, K), która uruchomi procesy w stosownym cyklu tak, by całość wypisywała coraz dłuższe listy; K to liczba stanowiąca jedyny element listy, od której rozpoczyna się działanie cyklu.

Przykładowo, wywołanie start$(N, 1)$, z dowolnym $N \geqslant 3$, powinno spowodować wypisanie następujących list:

```
[1]
[1,1]
[2,1,1]
[4,2,1,1]
...
```