Programowanie funkcyjne — kolokwium nr 2, 24.01.2024

Instrukcja: Każde zadanie należy przesłać na Pegaza w oddzielnym pliku: zadanie1.hs, zadanie2.hs i zadanie3.erl. Plików nie należy zipować. Rozwiązania muszą się poprawnie kompilować. 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. Napisać interaktywny program udający wszechwiedzącą sztuczną inteligencję, odpowiadającą na pytania tak/nie. Program powinien poprosić o pytanie, wczytać je ze standardowego wejścia, a następnie odpowiedzieć (na standardowe wyjście) twierdząco, jeśli w pytaniu znajdowało się parzyście wiele samogłosek, oraz przecząco w przeciwnym przypadku. Można założyć, że pytanie składa się tylko z małych i dużych liter (bez polskich znaków), liczb, przecinków i pytajników.

Zadanie 2. Jedną z możliwych definicji liczby Eulera jest suma szeregu

$$\sum_{i=0}^{\infty} \frac{1}{i!}.$$

Napisać bezpunktowo funkcję o sygnaturze

`liczbaEulera :: Int -> Double`,

przybliżającą liczbę Eulera poprzez sumę

$$\sum_{i=0}^{n} \frac{1}{i!}$$

dla nieujemnego argumentu n. W rozwiązaniu należy w istotny sposób użyć funkcji foldl/foldr. Uwaga: można zdefiniować pomocnicze funkcje, jeśli również są napisane bezpunktowo. Listę $[1..n]$ można otrzymać poprzez take n $[1..]$.

Zadanie 3. Napisać moduł tworzący w następujący sposób M-arne drzewo komunikujących się procesów o wysokości N. Liczby M, N i X są naturalne, większe od zera.

- Pierwszy proces (stanowiący korzeń drzewa na głębokości 0), wywołany z parametrem X, tworzy M procesów i przekazuje im jako parametr liczby MX, ..., $MX + M - 1$. Następnie czeka na komunikaty od wszystkich potomków, konkatenuje je i dokłada X jako głowę oraz wypisuje tak utworzoną listę na ekran.
- Każdy następny proces:
  - jeżeli jest na głębokości mniejszej niż N, to tworzy M procesów podobnie jak proces pierwszy, czeka na komunikaty, konkatenuje je dokładając X jako głowę, a listę wypisuje na ekran i odsyła do rodzica,
  - jeżeli jest na głębokości N, to wypisuje swój parametr X na ekran i odsyła do rodzica jako jednoelementową listę.

- Funkcja start(M, N, X) tworzy pierwszy proces w taki sposób, by przekazać mu parametr X i doprowadzić do zbudowania drzewa w opisany powyżej sposób.
- Procesy mogą otrzymywać jako parametry nie tylko liczby X; nie mogą jednak otrzymywać żadnych list.

Przykładowo, wywołanie start(2, 2, 1) powinno spowodować wypisanie następujących list: [4], [5], [6], [7], [2,4,5], [3,6,7], [1,2,4,5,3,6,7]. Kolejność list może być inna, zależnie od kolejności dostarczania komunikatów i dostępu do procesora.