

## Programowanie funkcyjne — kolokwium nr 2, 25.01.2023

**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 do gry w „papier, kamień, nożyce”. Komputer ma wykonywać serię kolejnych ruchów zgodnie z kolejnymi znakami stringu, podanego jako parametr podczas uruchomienia programu. Można założyć, że string zawiera dodatnią liczbę znaków ze zbioru {P, K, N}. Uruchomienie programu ma wykonać tyle rund gry, ile znaków jest w stringu; w każdej rundzie program prosi o wpisanie naszego ruchu (tj. znaku P, K lub N), po czym wypisuje swoją odpowiedź (na podstawie stringu) i bieżący stan rozgrywki (liczbę całkowitą, oznaczającą bilans rozgrywki na rzecz komputera), a następnie przechodzi do kolejnej rundy. Po wyczerpaniu ruchów program kończy działanie.

**Zadanie 2.** Napisać bezpunktowo funkcję o sygnaturze

`dodajPunkty :: Integer -> [[Integer]] -> [Integer]`,

dodając do siebie listę punktów z przestrzeni  $n$ -wymiarowej o współrzędnych całkowitych. Punkty reprezentowane są jako listy długości  $n$  (gdzie  $n$  to pierwszy argument funkcji), zatem lista punktów jest typu `[[Integer]]` (drugi argument funkcji). W rozwiązaniu należy w istotny sposób użyć funkcji `foldl` lub `foldr`. Można założyć, że na liście jest przynajmniej jeden punkt. Wskazówka: można użyć `replicate` i/lub `zipWith`.

**Zadanie 3.** Napisać moduł tworzący drzewo binarne, złożone z  $2^{N+1} - 1$  komunikujących się procesów, przy czym  $N \geq 1$ . Drzewo należy stworzyć w następujący sposób:

- Pierwszy proces (stanowiący korzeń drzewa na głębokości 0), wywołany z parametrem  $X$ , tworzy dwa procesy i przekazuje im jako parametr liczby  $2X$  i  $2X + 1$ . Następnie czeka na komunikaty od obydwu 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 dwa procesy podobnie jak proces pierwszy, czeka na komunikaty, konkatenuje je dokładając  $X$ , 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(N, X)` tworzy pierwszy proces w taki sposób, by przekazać mu parametr  $X$  i doprowadzić do zbudowania drzewa o wysokości  $N$ .
- Procesy mogą otrzymywać jako parametry nie tylko liczby  $X$ .

Przykładowo, wywołanie `start(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.