Paradygmaty programowania - ćwiczenia
Lista 1

Wszystkie funkcje należy napisać w obu językach: OCaml i Scala. W specyfikacjach jest wykorzystywana notacja OCamla, ponieważ jest prostsza. W programach można używać wyłącznie konstrukcji językowych przedstawionych na wykładach!

1. Zdefiniuj funkcję flatten : 'a list list -> 'a list, która dla argumentu będącego listą list tworzy listę, złożoną z elementów wszystkich podlist z zachowaniem ich kolejności, np. flatten [[5;6];[1;2;3]] zwraca [5; 6; 1; 2; 3], czyli spłaszcza listę o jeden poziom.

2. Zdefiniuj funkcję count : 'a * 'a list -> int obliczającą ile razy dany obiekt występuje w danej liście, np. count ('a', ['a'; 'l'; 'a']) zwraca 2.

3. Zdefiniuj funkcję replicate: 'a * int -> 'a list powtarzającą dany obiekt określoną liczbę razy i zwracającą wynik w postaci listy, np. replicate ("la",3) zwraca ["la"; "la"; "la"].

4. Zdefiniuj funkcję sqrList : int list -> int list podnoszącą do kwadratu wszystkie elementy danej listy liczb, np. sqrList [1;2;3;-4] zwraca [1; 4; 9; 16].

5. Zdefiniuj funkcję palindrome : 'a list -> bool sprawdzającą, czy dana lista jest palindromem, tj. równa się sobie samej przy odwróconej kolejności elementów, np. palindrome ['a'; 'l'; 'a'] zwraca true.

6. Zdefiniuj swoją funkcję listLength : 'a list -> int, obliczającą długość dowolnej listy (oczywiście bez użycia standardowej funkcji List.length).

7. Rozwiąż układ równań rekurencyjnych (zakładając, że N jest potęgą dwójki):

$$T(1) = 1$$
$$T(N) = c(\lg N) + T(N/2) \qquad \text{dla } N \geq 2$$

Wykorzystaj technikę zilustrowaną na wykładzie 1 (Dodatek: Złożoność obliczeniowa. Podstawowe pojęcia).