

Paradygmaty programowania - ćwiczenia

Lista 3

W zadaniach 2, 3, 5 funkcje należy napisać w obu językach: OCaml i Scala (wykorzystując mechanizm dopasowania do wzorca!).

1. Podaj (i wyjaśnij!) typy poniższych funkcji (samodzielnie, bez pomocy kompilatora OCaml!):
a) `let f1 x = x 1 1;;` b) `let f2 x y z = x (y ^ z);;`
2. Zdefiniuj funkcje a) *curry3* i b) *uncurry3*, przeprowadzające konwersję między zwinionymi i rozwiniętymi postaciami funkcji od trzech argumentów. Podaj ich typy.
3. Przekształć poniższą rekurencyjną definicję funkcji *sumProd*, która oblicza jednocześnie sumę i iloczyn listy liczb całkowitych na równoważną definicję nierekurencyjną z jednokrotnym użyciem funkcji bibliotecznej *fold_left* (Scala – *foldLeft*), której argumentem jest odpowiednia funkcja anonimowa (literal funkcyjny).

OCaml	Scala
<pre>let rec sumProd l = match l with h::t -> let (s,p)= sumProd t in (h+s,h*p) [] -> (0,1);;</pre>	<pre>def sumProd(xs:List[Int]):(Int,Int) = xs match { case h::t => {val (s,p)=sumProd(t) (h+s,h*p) } case Nil => (0,1) }</pre>

4. Poniższe dwie wersje funkcji *quicksort* działają niepoprawnie. Dlaczego?

OCaml

a) `let rec quicksort = function`
 `[] -> []`
 `[x] -> [x]`
 `xs -> let small = List.filter (fun y -> y < List.hd xs) xs`
 `and large = List.filter (fun y -> y >= List.hd xs) xs`
 `in quicksort small @ quicksort large;;`

b) `let rec quicksort' = function`
 `[] -> []`
 `x::xs -> let small = List.filter (fun y -> y < x) xs`
 `and large = List.filter (fun y -> y > x) xs`
 `in quicksort' small @ (x :: quicksort' large);;`

5. Zdefiniuj funkcje sortowania

a) przez wstawianie z zachowaniem stabilności i złożoności $O(n^2)$

insertionsort : ('a->'a->bool) -> 'a list -> 'a list .

b) przez łączenie (scalanie) z zachowaniem stabilności i złożoności $O(n \lg n)$

mergesort : ('a->'a->bool) -> 'a list -> 'a list .

Pierwszy argument jest funkcją, sprawdzającą porządek. Podaj przykład testu sprawdzającego stabilność.

Uwaga! Przypominam, że funkcje *List.append* i *List.length* mają złożoność liniową!