Lista 3

1. Napisz funkcje curry3 i uncurry3, przeprowadzające konwersje między zwiniętymi i rozwiniętymi postaciami funkcji od trzech argumentów.
   a) Użyj w definicjach możliwie dużo lukru syntaktycznego.
   b) Nie używaj w definicjach lukru syntaktycznego.

2. Napisz funkcję czylstnieje: ('a -> bool) -> 'a list -> bool. *czylstnieje* p xs ma zwracać wartość logiczną zdania „$\exists x \in xs.p(x)$"
   np.: czylstnieje (function x -> x=2) [1;2;3;5] => true.
   Należy napisać trzy wersje tej funkcji:
   a) z wykorzystaniem dopasowania do wzorca,
   b) z wykorzystaniem funkcjonału fold_left,
   c) z wykorzystaniem funkcjonału fold_right.

3. Napisz funkcjonał filter wykorzystując funkcjonał List.fold_right.

4. Napisz funkcję usun1: ('a -> bool) -> 'a list ->'a list; usun1 pred xs zawiera te same wartości, co lista xs, z której usunięto pierwszy element spełniający predykat pred.
   np.: usun1 (function x -> x=2) [1;2;3;2;5] => [1;3;2;5] .
   Należy napisać dwie wersje tej funkcji:
   a) ze zwykłą rekursją,
   b) z możliwie efektywną rekursją ogonową (użyj List.rev_append).

5. Napisz funkcję sortowania przez łączenie (scalanie zstępujące) z zachowaniem stabilności i złożoności $O(n \lg n)$
   mergesort: ('a->'a->bool) -> 'a list -> 'a list
   Pierwszy parametr jest funkcją, sprawdzającą porządek.
   Jeden z testów musi sprawdzać stabilność!