Programowanie funkcyjne i współbieżne
Lista 3

1. Napisz funkcję exists[A] (xs: List[A]) (p: A => Boolean): Boolean.
   exists (xs) (p) ma wartość logiczną zdania „$\exists x \in xs.p(x)$”
   np. exists (List(5,1,2,3)) (_ == 2)

   Należy napisać trzy wersje tej funkcji:
   a) z wykorzystaniem dopasowania do wzorca i rekursji,
   b) z wykorzystaniem metody List.foldLeft,
   c) z wykorzystaniem metody List.foldRight.

2. Napisz funkcję filter[A](xs: List[A])(p: A => Boolean): List[A] wykorzystując funkcjonał List.foldRight.
   np. filter (List(2,7,1,3,7,8,4,1,6,9)) (_ > 3) == List(7, 7, 8, 4, 6, 9)

3. Napisz funkcję remove1[A](xs: List[A])(p: A => Boolean): List[A] zwracający listę z tymi samymi wartościami, co lista xs, z której usunięto pierwszy element spełniający predykat p.
   np. remove1(List(1,2,3,2,5)) (_ == 2) == List(1, 3, 2, 5)

   Należy napisać dwie wersje tej funkcji:
   a) ze zwykłą rekursją,
   b) z możliwie efektywną rekursją ogonową (użyj List.reverse_:::).

4. Napisz funkcję splitAt[A](xs: List[A])(n: Int): (List[A], List[A]), zwracającą parę równą (xs take n, xs drop n), ale bez dwukrotnego przechodzenia listy xs.
   np. splitAt (List('a','b','c','d','e')) (2) == (List('a', 'b'), List('c', 'd', 'e'))