Programowanie funkcyjne i współbieżne
Lista 2

W funkcjach 1-4 należy wykorzystywać mechanizm dopasowania do wzorca!
Funkcje 1-3 mają mieć złożoność liniową względem długości listy wejściowej.

1. Napisz funkcję take[A](n: Int, xs: List[A]): List[A],
   gdzie take(k, List(x_1, ..., x_n)) == List(x_1, ..., x_k),
   np. take(2, List(1,2,3,5,6)) == List(1,2)
   take(-2, List(1,2,3,5,6)) == Nil
   take(8, List(1,2,3,5,6)) == List(1,2,3,5,6)

2. Napisz funkcję drop[A](n: Int, xs: List[A]): List[A],
   gdzie drop(k, List(x_1 , ..., x_n)) == List(x_{k+1}, ..., x_n),
   np. drop(2, List(1,2,3,5,6)) == List(3,5,6)
   drop(-2, List(1,2,3,5,6)) == List(1,2,3,5,6)
   drop(8, List(1,2,3,5,6)) == Nil

3. Napisz funkcję reverse[A](xs: List[A]): List[A], odwracającą zadaną listę w czasie liniowym (bez użycia metody bibliotecznej reverse!),
   np. reverse(List("Ala", "ma", "kota")) == List("kota", "ma", "Ala")

4. Napisz funkcję replicate: List[Int] => List[Int], która z danej listy liczb naturalnych tworzy listę, w której każdy element wejściowej listy jest tyle razy powtórzony, jaką ma wartość,
   np. replicate (List(1,0,4,-2,3)) == List(1, 4, 4, 4, 4, 3, 3, 3)

5. Dla zadanej liczby rzeczywistej a oraz dokładności ε można znaleźć pierwiastek trzeciego stopnia z a wyliczając kolejne przybliżenia x_i tego pierwiastka (metoda Newtona-Raphsona):

   $x_0 = a/3$ dla $a > 1$
   $x_0 = a$ dla $a \leq 1$
   $x_{i+1} = x_i + (a/x_i^2 - x_i)/3$

   Dokładność (względna) jest osiągnięta, jeśli $|x_i^3 - a| \leq \varepsilon * |a|$.

   Napisz efektywną (**wykorzystującą rekursję ogonową**) funkcję root3: Double => Double, która dla zadanej liczby a znajduje pierwiastek trzeciego stopnia z dokładnością względną $\varepsilon = 10^{-15}$.