Semnături funcții haskell

```
add :: Integer -> Integer -> Integer
inc = add 1

map :: (a->b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x : (xs ++ ys)

(.) :: (b->c) -> (a->b) -> (a->c)
f . g = \x -> f (g x)

ones = 1 : ones

nTimes f 0 x = x
nTimes f n x = f (nTimes f (n-1) x)

squares = map (^2) (nTimes (+2) 0)

fib = 1 : 1 : [a+b | (a,b) <- zip fib (tail fib)]

zip (x:xs) (y:ys) = (x,y) : zip xs ys
zip xs ys = []

head (x:xs) = x
head [] = error "head"; head :: [a] -> a

length :: [a] -> Integer
length [] = 0
length (x:xs) = 1 + length xs

tail :: [a] -> [a]
tail (x:xs) = xs
```

x `elem` [] = False

x `elem` (y:ys) = x == y || (x `elem` ys)

elem :: (Eq a) => a -> [a] -> Bool

(+), (-), (*) :: (Num a) => a -> a -> a

negate, abs :: (Num a) => a -> a.

truncate, round, floor, ceiling, numerator, denominator, conjugate, head, sqrt,

take, drop, filter