**Laboratorul 1**

Ex 1

squareSum :: Natural -> Natural -> Natural

let squareSum = \x -> \y -> (+ (\* x x) (\* y y)) in squareSum 2 3

Ex 2

revRange :: Natural -> [Natural]

letrec revRange = \x -> if(isZero x) [] (++ [(- x 1)] (revRange (pred x))) in revRange 4

letrec revRange = (\x -> if (isZero x) [] (++ [(- x 1)] (revRange (pred x)))) in let range = (\x -> (reverse (revRange x))) in range 4

Ex 3

justList :: [Maybe Natural] -> [Natural]

let justList = \x ->map (fromMaybe 0) (filter (isJust) x) in justList [Just 4, Nothing, Just 5, Just 7, Nothing]

Ex 4

all :: (Natural -> Bool) -> [Natural] -> Bool

let all = \f -> \l -> foldr (&&) True (map f l) in all isZero [0, 0, 0]

let all = \f -> \l -> foldr (&&) True (map f l) in all isZero [0, 1]