1 Velg/angi riktige svar

1.1 Evaluering av `filter even (map (*2) [1 .. 5])` gir:

a) `[2,4]`

b) `[4,8]`

c) `[2,6,10]`

d) `[2,4,6,8,10]`

Svar. Funksjonen `filter even (map (*2) [1 .. 5])` vil først multiplisere alle verdiene i listen med 2, og vi får listen `[2,4,6,8,10]`. `filter even` vil filtrere vekk alle verider som ikke er partall, og vi vil til slutt ende opp med listen `[2,4,6,8,10]`.

1.2 Hva blir resultatet av å evaluere `take 5 nats` med følgende definisjoner?

a) `nats = 0:1:tail nats`

b) `nats = 0 : tail nats`

c) `nats = 0:map (+1) nats`

d) `nats = map (+1) [0..]`

Svar. a) `nats = 0:1:tail nats = [0,1,1,1,1]`

b) `nats = 0 : tail nats = [0`

c) `nats = 0:map (+1) nats = [0,1,2,3,4]`

d) `nats = map (+1) [0..] = [1,2,3,4,5]`

1.3 Hva blir resultatet av å evaluerere `concat ["ab", "cd", "","efg"]` med hver av følgende defininsjoner?

a) `concat xss = [x | x <- xss]`

b) `concat xss = [x | xs <- xss, x <- xs]`

c) `concat xss = concat (tail xss)`

d) `concat xss = map (++) xss`

Svar. a) `concat xss = [x | x <- xss]` = [”ab”,”cd”,””,”efg”]

b) `concat xss = [x | xs <- xss, x <- xs] = abcdefg`

c) `concat xss = concat (tail xss) = cdefg`

d) `concat xss = map (++) xss = ["bc","de", "", "fgh"]`

1.4 Funksjonen `apply` definert ved `apply f x = f x` har typen

a) `a -> b -> c`

b) `(a -> b) a -> b`

c) `a -> (b -> a) -> b`

d) `a -> b -> (a -> b)`

Svar. Denne funksjonen har typen `(a -> b) -> a -> b`

2 Matrisemultiplikasjon

Svar. Kode for Matrisemultiplikasjon:

```
import Data.List (transpose)

row :: [[Int]] -> Int -> [Int]
row m n = m !! max 0 (n - 1)

col :: [[Int]] -> Int -> [Int]
col m n = transpose m !! max 0 (n - 1)

cols :: [[Int]] -> [[Int]]
cols = transpose

mult :: [[Int]] -> [[Int]] -> [[Int]]
mult m n = [map (multrc (row m x)) (cols n) | x <- [1 .. length m]]

multrc :: [Int] -> [Int] -> Int
multrc r c = sum [x * y | (x,y) <- zip r c]
```