JoonHeeOoten_Assignment7 > Assignment7.hs

```haskell
-- replicate' function
replicate' :: Int -> a -> [a]
replicate' n x = [x | _ <- [1..n]]

-- perfects function
perfects :: Int -> [Int]
perfects n = [x | x <- [1..n], x == sum (factors x)]

-- factors function
factors :: Int -> [Int]
factors n = [x | x <- [1..n-1], n `mod` x == 0]

-- find function
find :: Eq a => a -> [(a, b)] -> [b]
find key table = [val | (k, val) <- table, k == key]

-- positions function
positions :: Eq a => a -> [a] -> [Int]
positions x xs = find x (zip xs [0..])

-- scalarproduct function
scalarproduct :: [Int] -> [Int] -> Int
scalarproduct xs ys = sum [x * y | (x, y) <- zip xs ys]

-- Test cases and screen prints
main :: IO ()
main = do
    putStrLn "replicate' function:"
    print $ replicate' 5 "test code"

    putStrLn "\nperfects function:"
    print $ perfects 9000

    putStrLn "\nfind function:"
    print $ find 'c' [('a', 1), ('b', 2), ('c', 3), ('b', 4), ('c', 25)]

    putStrLn "\npositions function:"
    print $ positions 1 [1, 0, 0, 1, 0, 1, 1, 0]

    putStrLn "\nscalarproduct function:"
    print $ scalarproduct [-1, 2, 3] [-4, -5, 6]

```

Terminal

```
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking main ...
replicate' function:
["test code","test code","test code","test code","test code"]

perfects function:
[6,28,496,8128]

find function:
[3,25]

positions function:
[0,3,5,6]

scalarproduct function:
12
```