# List Replication

**Clojure**

(fn [num lst]

(let[ x (flatten (map (fn [e] (repeat num e)) lst))]

(doseq [i x] (println i)))

)

**Racket**

#lang racket

(define (intlist)

(let ([new (read)])

(if (eof-object? new)

(list)

(cons (number->string new) (intlist)))))

(define (change n s)

(string-append\* (make-list n (string-append s "\n"))))

(define t (read))

(define xs (intlist))

(define ys (map (lambda (new) (change t new)) xs))

(for ([new ys])

(printf "~a" new))

**Scala**

def f(num:Int,arr:List[Int]):List[Int] = arr.flatMap(List.fill(num)(\_))

**flatMap link:** <https://www.brunton-spall.co.uk/post/2011/12/02/map-map-and-flatmap-in-scala/>

# Filter Array

**Scala**

def f(n:Int, arr:List[Int]) = {

arr.filter(\_<n)

}

**.filter link:** <https://alvinalexander.com/scala/how-to-use-filter-method-scala-collections-cookbook>

Clojure

(fn[delim lst] (

filter #(< % delim) lst

))

**Haskell**

f :: Int -> [Int] -> [Int]

f = filter . (>)

-- The Input/Output section. You do not need to change or modify this part

main = do

n <- readLn :: IO Int

inputdata <- getContents

let

numbers = map read (lines inputdata) :: [Int]

putStrLn . unlines $ (map show . f n) numbers

# Filter Positions in a List

**Scala**

def f(arr:List[Int]):List[Int] = {

val l = scala.collection.mutable.ListBuffer.empty[Int]

val range=arr.indices.filter { x => x%2!=0 }

range.foreach { x => l += arr(x) }

l.toList

}

**.foreach link:** <https://alvinalexander.com/scala/iterating-scala-lists-foreach-for-comprehension>

**.tolist link:** <https://alvinalexander.com/scala/list-class-methods-examples-syntax>

**Haskell**

f :: [Int] -> [Int]

f xs = go 1 xs

where go \_ [] = []

go n (x:xs) = if (mod n 2) == 0 then x:(go (n+1) xs) else go (n+1) xs

-- This part deals with the Input and Output and can be used as it is. Do not modify it.

main = do

inputdata <- getContents

mapM\_ (putStrLn. show). f. map read. lines $ inputdata

**Lambda Calculus - Reductions #4**

CAN'T REDUCE

# Lambda Calculus - Evaluating Expressions #5

0