```haskell
module Haskell_Mini_Project
( prefix
, reduce
, scatter
, gather
) where

import           Assist_Lib

-- 1 --
prefix :: (a -> a -> a) -> [a] -> [a]
prefix = scanl1'

scanl' _ _ [] = []
scanl' func curr (x:xs) = new : scanl' func new xs
  where new = func curr x

scanl1' func (x:xs) = x : scanl' func x xs

-- 2 --
reduce :: (Num a) => (a -> a -> a) -> a -> [a] -> a
reduce func num xs = foldl1 (func) $ num : xs

-- 3 --
scatter :: (Integral a) => a -> [b] -> [(a, [b])]
scatter numberOfCores sharableData =
      combine possibleCores $ distributedData numberOfCores sharableData
  where
    possibleCores = [0..numberOfCores-1]
    distributedData numberOfCores = zip (cycle possibleCores)

combine [] _ = []
combine (coreNum:remainingCores) [] = (coreNum, []) : combine remainingCores []
combine (coreNum:remainingCores) all@(x:xs) =
      grabAllData coreNum all :
            combine remainingCores (removeExtraData coreNum xs)

grabAllData key theData = (key, [snd tuple | tuple <- theData, fst tuple == key])
removeExtraData key = filter (\x -> fst x /= key)

-- 4 --
gather :: (Num a) => [(a, [b])] -> (a, [b])
gather allDistributedData = (numCores, recombineData allDistributedData)
  where
    numCores = fst (last allDistributedData) + 1
    recombineData [] = []
    recombineData (first:allTheRest) = snd first ++ recombineData allTheRest
```