```
histogram :: Map Int (Bag word) → Map word Int
histogram = mapReduce groupOnBags additiveGroupOnIntegers histogramMap histogramReduce
  where additiveGroupOnIntegers = Group (+) (λn → −n) 0
        histogramMap _          = foldBag groupOnBags (λn → singletonBag (n, 1))
        histogramReduce _       = foldBag additiveGroupOnIntegers id

-- Precondition:
-- For every key1 :: k1 and key2 :: k2, the terms mapper key1 and reducer key2 are homomorphisms.
mapReduce :: Group v1 → Group v3 → (k1 → v1 → Bag (k2, v2)) → (k2 → Bag v2 → v3) →
             Map k1 v1 → Map k2 v3
mapReduce group1 group3 mapper reducer = reducePerKey ∘ groupByKey ∘ mapPerKey
  where mapPerKey    = foldMap group1 groupOnBags mapper
        groupByKey   = foldBag  (groupOnMaps groupOnBags)
                                (λ(key, val) → singletonMap key (singletonBag val))
        reducePerKey = foldMap groupOnBags (groupOnMaps group3)
                                (λkey bag    → singletonMap key (reducer key bag))
```