```
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 key_1 :: k_1 and key_2 :: k_2, the terms mapper key_1 and reducer key_2 are homomorphisms.
mapReduce :: Group v_1 → Group v_3 → (k_1 → v_1 → Bag (k_2, v_2)) → (k_2 → Bag v_2 → v_3) → Map k_1 v_1 → Map k_2 v_3
mapReduce group_1 group_3 mapper reducer = reducePerKey ∘ groupByKey ∘ mapPerKey
  where mapPerKey    = foldMap group_1 groupOnBags                                  mapper
        groupByKey   = foldBag  (groupOnMaps groupOnBags)                           (λ(key, val) → singletonMap key (singletonBag val))
        reducePerKey = foldMap groupOnBags (groupOnMaps group_3) (λkey bag    → singletonMap key (reducer key bag))
```