Similarly, we can create a set that supports removals by combining two G-Sets

We call it a **Two-phase-Set**

2P-Set = A G-Set for adds + a G-Set for removals, which we call a *tombstone set*

The members of the 2P-Set are elements of the first set that don’t appear in the tombstone set

Use case: count active users on a P2P network

Similarly, we can create a set that supports removals
by combining two G-Sets

We call it a **Two-phase-Set**

2P-Set = A G-Set for adds + a G-Set for removals,
which we call a *tombstone set*

The members of the 2P-Set are elements of the first set that
don’t appear in the tombstone set

Use case: count active users on a P2P network

```haskell
-- | A set that supports adding elements, or removing it definitively
-- | Removed elements are kept around in a special *tombstone* set.
data TwoPhaseSet t
  = TwoPhaseSet (GSet.GSet t) (GSet.GSet t)

insert :: forall t. Hashable t => t -> TwoPhaseSet t -> TwoPhaseSet t
insert value (TwoPhaseSet a b) = TwoPhaseSet (GSet.insert value a) b

remove :: forall t. Hashable t => t -> TwoPhaseSet t -> TwoPhaseSet t
remove value (TwoPhaseSet a b) = TwoPhaseSet a $ GSet.insert value b

query :: forall t. Hashable t => TwoPhaseSet t -> Set.HashSet t
query (TwoPhaseSet a b) = Set.difference (GSet.query a) (GSet.query b)
```