```
module LocalStore {

  type    Ptr = nat
  type    Val = object?
  type  Store = map<Ptr,Val>

  method initially() returns (s : Store)
    ensures s == map[]
    { return map[]; }

  method add(s' : Store, v : Val, p : Ptr) returns (s : Store)
    requires p !in s'.Keys
    ensures  s == s'[p:=v]
    { s :=  s'[p:=v]; }

  method update(s' : Store, v : Val, p : Ptr) returns (s : Store)
    requires p in s'.Keys
    ensures  s == s'[p:=v]
    { s :=  s'[p:=v]; }

  method deref(s' : Store, p : Ptr) returns (v : Val)
    requires p in s'.Keys
    ensures  v == s'[p]
    { v := s'[p]; }

  method transfer(s' : Store, t' : Store, ptrs : set<Ptr> )
      returns (s : Store, t : Store)
    requires ptrs <= s'.Keys
    ensures  s == s' - ptrs
    ensures  t == t' + (s' - (s'.Keys - ptrs))
    {s := s' - ptrs; t := t' + (s' - (s'.Keys - ptrs)); }

}
```