<exp> := ...

(box <exp>)

(set-box <exp> <exp>)

(get-box <exp>)

(block <exp> <exp> ...)

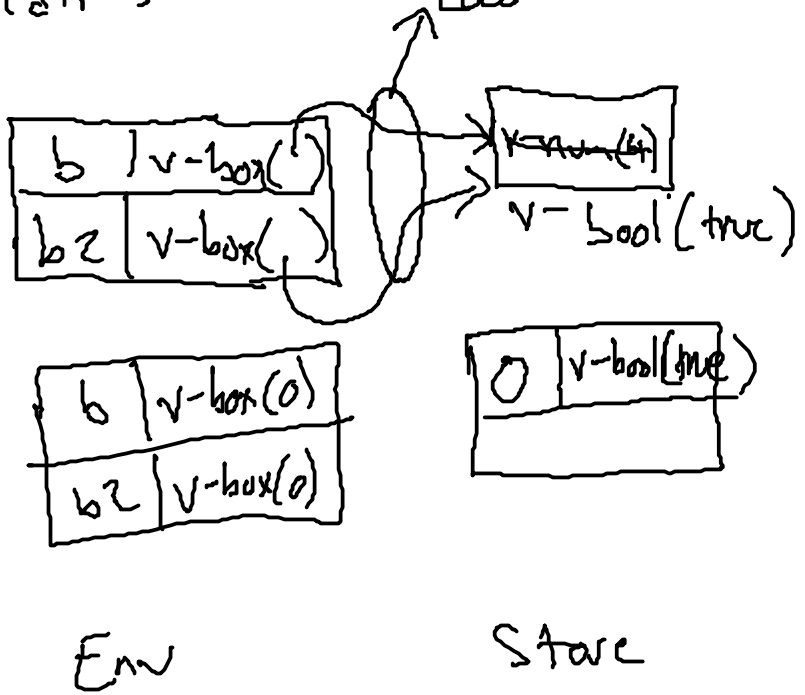
alloc :: Store, Value → Allocation

update :: Store, Value, Loc → Store

deref :: Store, Loc → Value

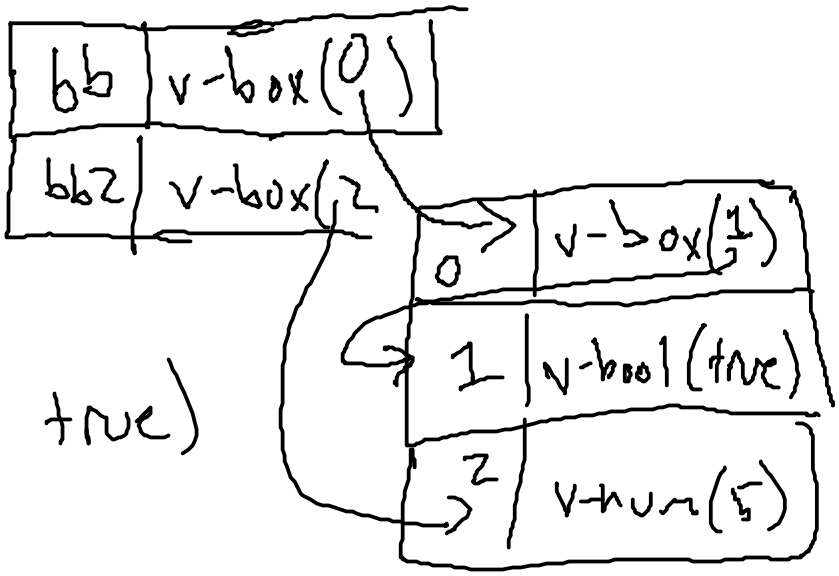
Locations = Number

```
(let (b (box 1))
  (let (b2 b)
    (block
      (set-box b2 true)
      (get-box b))))
```



```
(let (bb (box (box 4)))
  (block
    (get-box bb)
    (get-box bb)
    (set-box (get-box bb) true)
    (let (bb2 (box 5))
```

v-box(1)



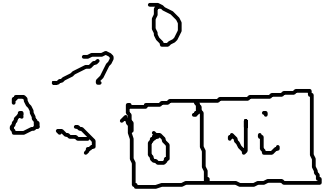

```
data Store:
  | mt-sto
  | sto(l :: Location,
       v :: Value,
       rest :: Store)

data Allocation
  | allocation(l :: Location, s :: Store)
end
```

```
fun alloc(s :: Store, v :: Value)
  -> Allocation:
  cases(Store) s:
    | mt-sto =>
      allocation(0, sto(0,
                        v,
                        mt-sto))
    | sto(l, _, _) =>
      allocation(l+1,
                 sto(l+1, v, s))
```

```
fun interp(env::Env, expr::Expr, store::Store) -> VandS
  cases(Expr) expr:
    ...
    | e-box(init) =>
      vi = interp(env, init, store)
      a = alloc(vi.s, vi.v)
      vs(v-box(a.l), a.s)
```

```
data VandS:
  | vs(v::Value,
       s::Store)
```

```
allocation(
    l::Loc
    s::Store
)
```




```
data Value:
  ...
  | v-box(
      l::Location)
```

```
var btns = [];
for (var i = 0; i < 10; i++) {
    btns[i] = function() { return int(i); }
}
```

i = i+1

```
btns[1]();  // 1   10
```


```
(let (f (let (x 5) (lam () x)))
  (f))
```