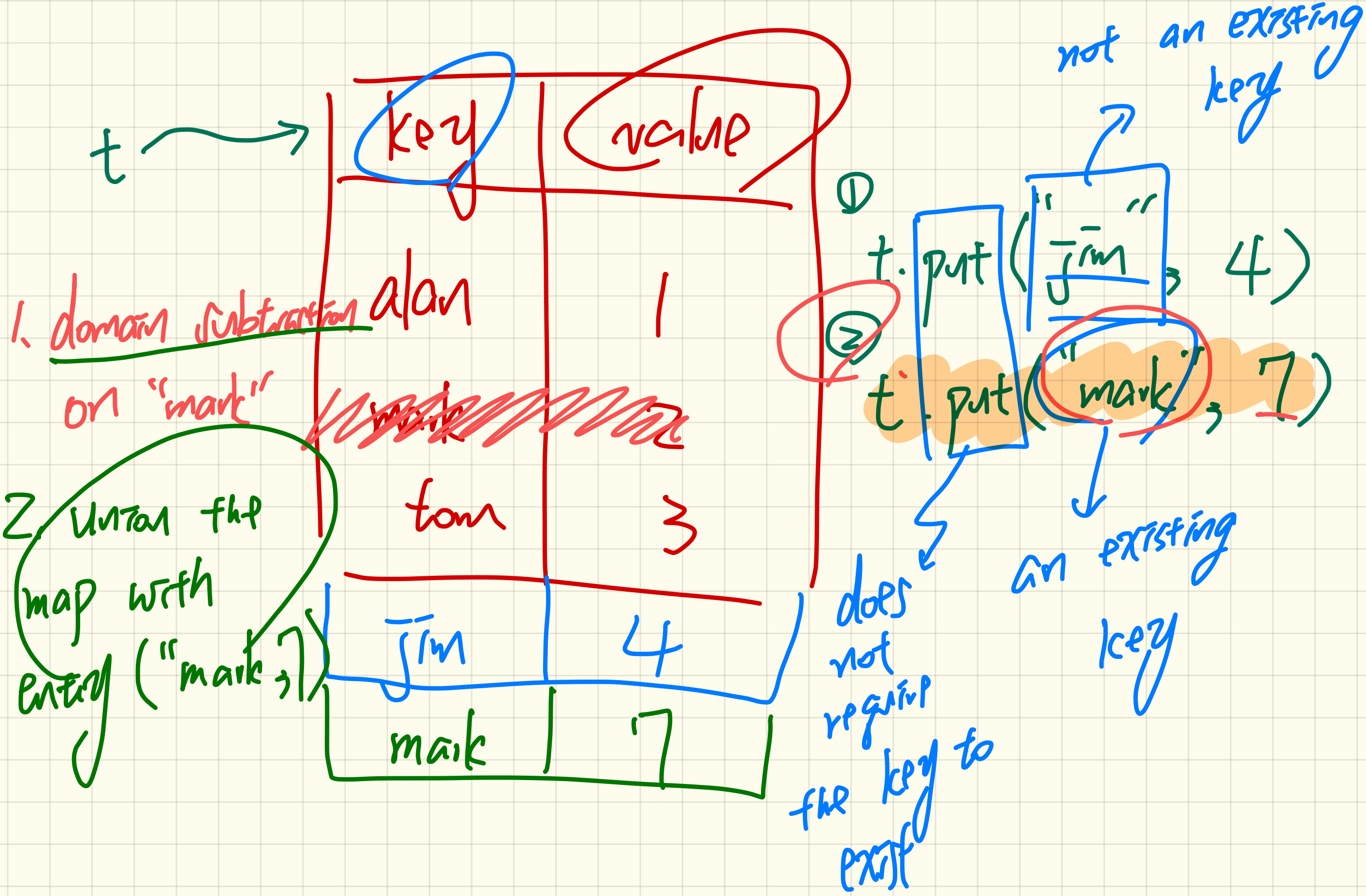
# LECTURE 9

MONDAY FEBRUARY 3

- **Labtest 1:**
  * Birthday Book
  * MATHMODELS
  * Iterator Patterns: Two Tutorial Series

REL FUN
PAIR



t

| key | value |
|---|---|
| alan | 1 |
| ~~mark~~ | ~~2~~ |
| tom | 3 |
| jim | 4 |
| mark | 7 |

1. domain subtraction on "mark"

2. union the map with entry ("mark", 7)

① t.put("jim", 4)

② t.put("mark", 7)

not an existing key

does not require the key to exist

an existing key

# Testing REL in MATHMODELS

REL : override cmd.

return new relation

$$r.\mathbf{overridden}(\underbrace{\{(a,3),(c,4)\}}_{t})$$

$$= \underbrace{\{(a,3),(c,4)\}}_{t} \cup \underbrace{\{(b,2),(b,5),(d,1),(e,2),(f,3)\}}_{r.\mathbf{domain\_subtracted}(\underbrace{t.\mathbf{domain}}_{\{a,c\}})}$$

$$= \{(a,3),(c,4),(b,2),(b,5),(d,1),(e,2),(f,3)\}$$

["a", 3]

```
test_rel: BOOLEAN
  local
    r, t: REL[STRING, INTEGER]
    ds: SET[STRING]
  do
    create r.make_from_tuple_array (
      <<["a", 1], ["b", 2], ["c", 3],
        ["a", 4], ["b", 5], ["c", 6],
        ["d", 1], ["e", 2], ["f", 3]>>)
    create ds.make_from_array (<<"a">>)
    -- r is not changed by the query 'domain_subtracted'
    t := r.domain_subtracted (ds)
    Result :=
      t /~ r and not t.domain.has ("a") and r.domain.has ("a")
    check Result end
    -- r is changed by the command 'domain_subtract'
    r.domain_subtract (ds)
    Result :=
      t ~ r and not t.domain.has ("a") and not r.domain.has ("a")
  end
```

(a, 3), (c, 4)

Say $r = \{\cancel{(a,1)},(b,2),\cancel{(c,3)},\cancel{(a,4)},(b,5),\cancel{(c,6)},(d,1),(e,2),(f,3)\}$

- r.***domain***: set of first-elements from *r*
  - r.**domain** = $\{\ d \mid (d,r) \in r\ \}$
  - e.g., r.**domain** = $\{a,b,c,d,e,f\}$
- r.***range***: set of second-elements from *r*
  - r.**range** = $\{\ r \mid (d,r) \in r\ \}$
  - e.g., r.**range** = $\{1,2,3,4,5,6\}$
- r.***inverse***: a relation like *r* except elements are in reverse order
  - r.**inverse** = $\{\ (r,d) \mid (d,r) \in r\ \}$
  - e.g., r.**inverse** = $\{(1,a),(2,b),(3,c),(4,a),(5,b),(6,c),(1,d),(2,e),(3,f)\}$
- r.***domain_restricted***(ds): sub-relation of *r* with domain *ds*.
  - r.**domain_restricted**(ds) = $\{\ (d,r) \mid (d,r) \in r \land d \in ds\ \}$
  - e.g., r.**domain_restricted**({a, b}) = $\{(\mathbf{a},1),(\mathbf{b},2),(\mathbf{a},4),(\mathbf{b},5)\}$
- r.***domain_subtracted***(ds): sub-relation of *r* with domain not *ds*.
  - r.**domain_subtracted**(ds) = $\{\ (d,r) \mid (d,r) \in r \land d \notin ds\ \}$
  - e.g., r.**domain_subtracted**({a, b}) = $\{(\mathbf{c},6),(\mathbf{d},1),(\mathbf{e},2),(\mathbf{f},3)\}$
- r.***range_restricted***(rs): sub-relation of *r* with range *rs*.
  - r.**range_restricted**(rs) = $\{\ (d,r) \mid (d,r) \in r \land r \in rs\ \}$
  - e.g., r.**range_restricted**({1, 2}) = $\{(a,\mathbf{1}),(b,\mathbf{2}),(d,\mathbf{1}),(e,\mathbf{2})\}$
- r.***range_subtracted***(ds): sub-relation of *r* with range not *ds*.
  - r.**range_subtracted**(rs) = $\{\ (d,r) \mid (d,r) \in r \land r \notin rs\ \}$
  - e.g., r.**range_subtracted**({1, 2}) = $\{(c,\mathbf{3}),(a,\mathbf{4}),(b,\mathbf{5}),(c,\mathbf{6})\}$

override ( s : SET<PAIR .. >)

overridden ( s: SET < PAIR .. >) : REL[G,H]

$\underline{\text{r. override ( s )}}$ → command does not return

↳ not to be used in contract

r. overridden ( s ) . domain

r. overridden(<<["a", 100], ["a", 200]>>)

Say $r = \{\cancel{(a,1)}, (b,2), (c,3), \cancel{(a,4)}, (b,5), (c,6), (d,1), (e,2), (f,3)\}$

$(a,100)$ $(a,200)$

relation

vs.

$$\begin{tikzcd} & r_1 \\ d_1 \arrow[ur] \arrow[dr] & \\ & r_2 \end{tikzcd}$$

function

$$\begin{tikzcd} d_1 \arrow[r] & r_1 \end{tikzcd}$$

```
all_positive_values (a: ARRAY[INTEGER]): ARRAY[INTEGER]
    require
        no_duplicates: ??
    ensure
        across Result is x
            all
                x > 0
            end
```

across | 1..| a.count is i all

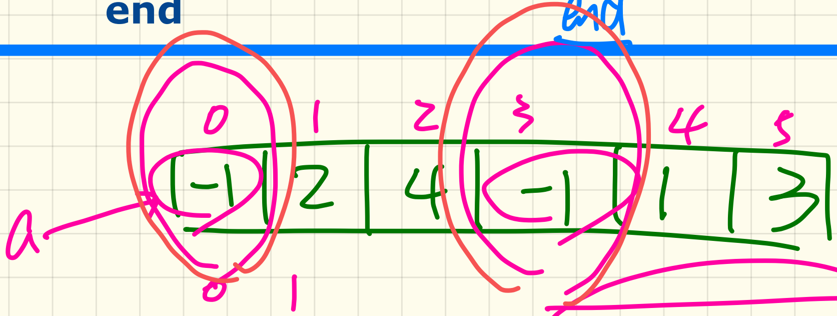
Boolean ← across |1..| a.count is j all

i+1

[ ]

end

end



0 1

0 2

0 3

⋮

~~a~~

0 ≠ 3

a[0] = a[3]

distinct locations implies

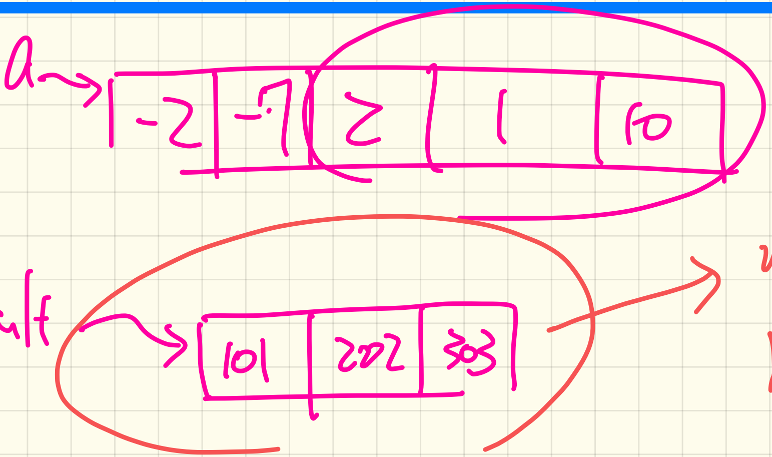
distinct values

# Writing Postcondition: Exercise

```
all_positive_values (a: ARRAY[INTEGER]): ARRAY[INTEGER]
    require
      no_duplicates: ??
    ensure
      across Result is x
        all
          x > 0
        end
```

all_p_v(<<-1, -7, 2, 1, 10>>)
↳ << 2, 1, 10 >>

Incomplete.

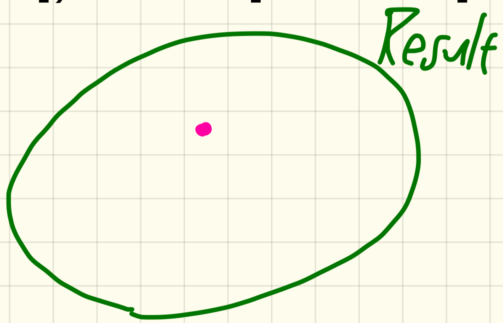


a → | -2 | -7 | 2 | 1 | 10 |

Result → | 101 | 202 | 303 |

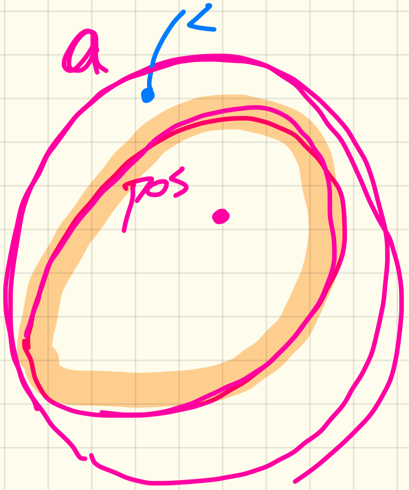
→ wrong imp. but postcond. evaluates to T.

```
all_positive_values (a: ARRAY[INTEGER]): ARRAY[INTEGER]
    require
        no_duplicates: ??
    ensure
        ~~across Result is x~~
            ~~all~~
                ~~x > 0~~
            ~~end~~
```



```
post_1: all_pos_in_a_also_in_result :
    across a is n all
        if n > 0 then
            Result.has(n)
        else
            ~~not Result.has(n)~~ True
        end
```

n > 0
implies
R.has(n)



```
all_positive_values (a: ARRAY[INTEGER]): ARRAY[INTEGER]
    require
        no_duplicates: ??
    ensure
        across Result is x
            all
                x > 0
        end
```

→ assume 'a' is not modified

```
across a is n all
    n > 0 implies Result.has(n)
end
```

incomplete

?

a → | 2 | ~~1~~ | 3 |

Result → | 2 | 3 | 44 |

should not be in!

$$\overset{\text{all pos. \#s in a}}{\boxed{S}} = \bigcirc\!\!\!\!\!\!T \rightarrow \text{Result.}$$

$$\equiv \underbrace{\boxed{S \subseteq T}}_{\text{each pos. \# in a is also in Result}} \wedge \underbrace{\boxed{T \subseteq S}}_{\text{each \# in Result is} \underset{\text{pos.}}{\wedge} \text{also in a.}}$$

```
all_positive_values (a: ARRAY[INTEGER]): ARRAY[INTEGER]
    require
        no_duplicates: ??
    ensure
        across Result is x
            all
                x > 0
            end
```

Witness

a → | -1 | 2 |

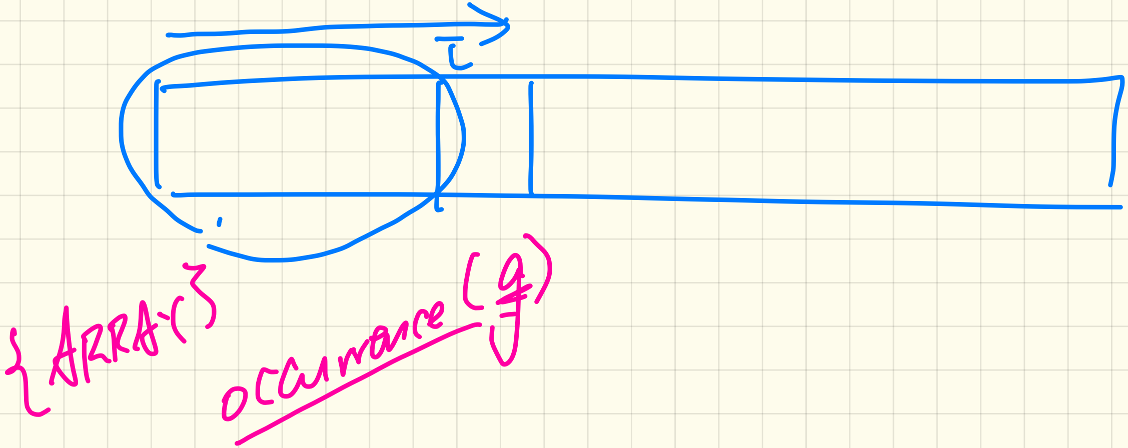
Result → | 2 | 2 |

```
all_pos_in_a_in_result :   across a is n all
                               n > 0 implies Result.has(n)
                           end

all_n_in_result_in_a :     across Result is n all
                               n > 0 and a.has(n)
                           end .
```

not complete

resolution : no_duplicates_in_result : ??

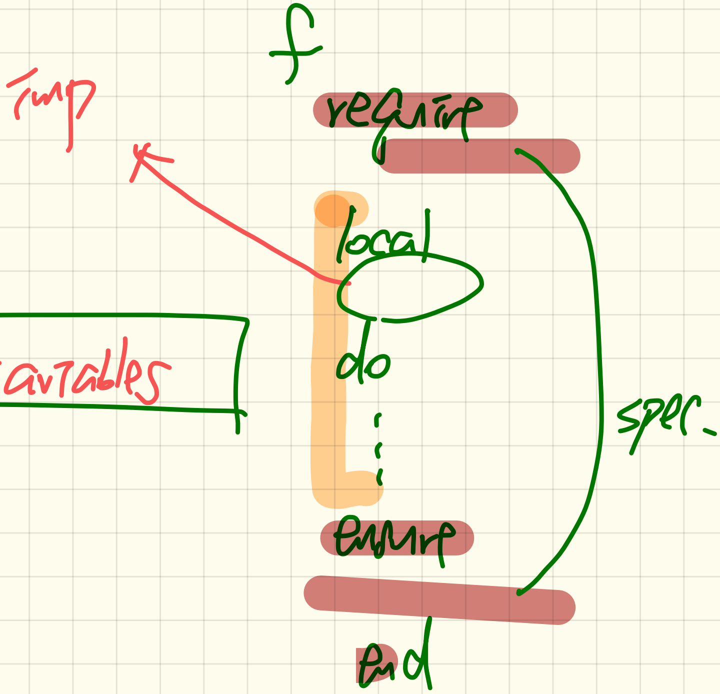

i
{ARRAY(}
occurrance (g)

require

- → attributes
- → quaries
- → X local variables
- → X old

ensure

- → attributes
- → quires
- → X local variable
- → ✓ old -

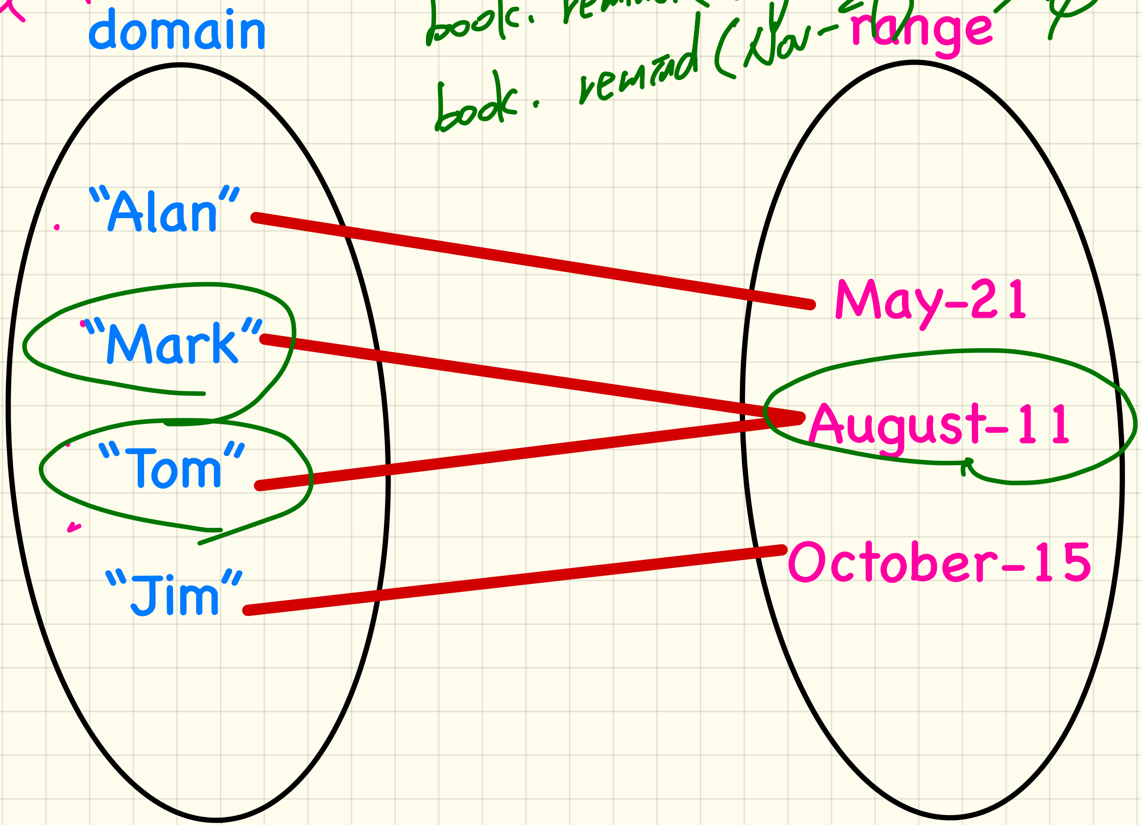


# Model of an Example Birthday Book

Count 4

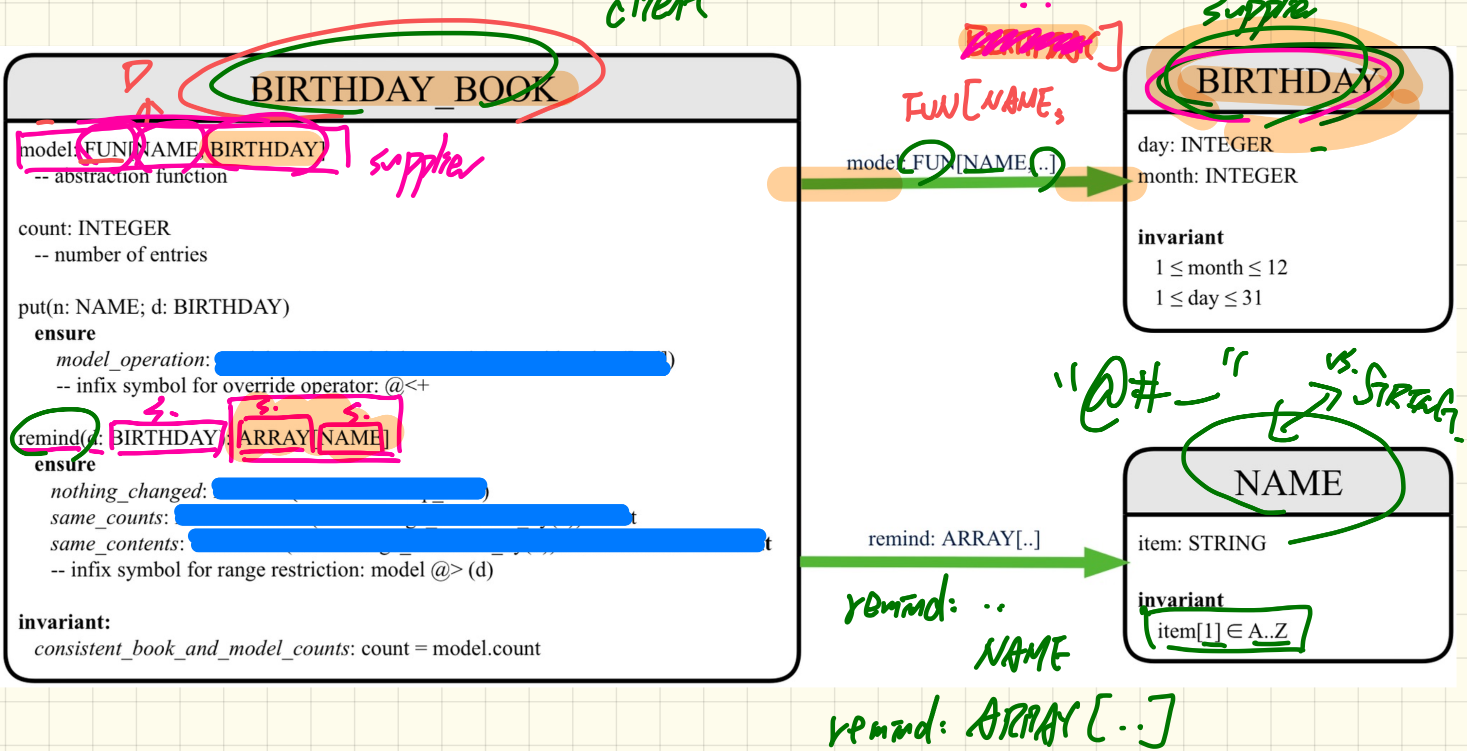
book.remind(August-11) → Mark, Tom

book.remind(Nov-29) → ∅



# Birthday Book: Design

client



BIRTHDAY_BOOK

D

model: FUN[NAME, BIRTHDAY]
-- abstraction function

supplier

count: INTEGER
-- number of entries

put(n: NAME; d: BIRTHDAY)
**ensure**
*model_operation*: [redacted]
-- infix symbol for override operator: @<+

remind(d: BIRTHDAY): ARRAY[NAME]
**ensure**
*nothing_changed*: [redacted]
*same_counts*: [redacted]
*same_contents*: [redacted]
-- infix symbol for range restriction: model @> (d)

**invariant:**
*consistent_book_and_model_counts*: count = model.count

~~BIRTHDAY~~]

FUN[NAME,

model: FUN[NAME, ..]

supplier

BIRTHDAY

day: INTEGER
month: INTEGER

**invariant**
1 ≤ month ≤ 12
1 ≤ day ≤ 31

"@#_" vs. STRING

remind: ARRAY[..]

remind: ..
NAME

NAME

item: STRING

**invariant**
item[1] ∈ A..Z

remind: ARRAY[..]