LECTURE 24

MONDAY MARCH 30

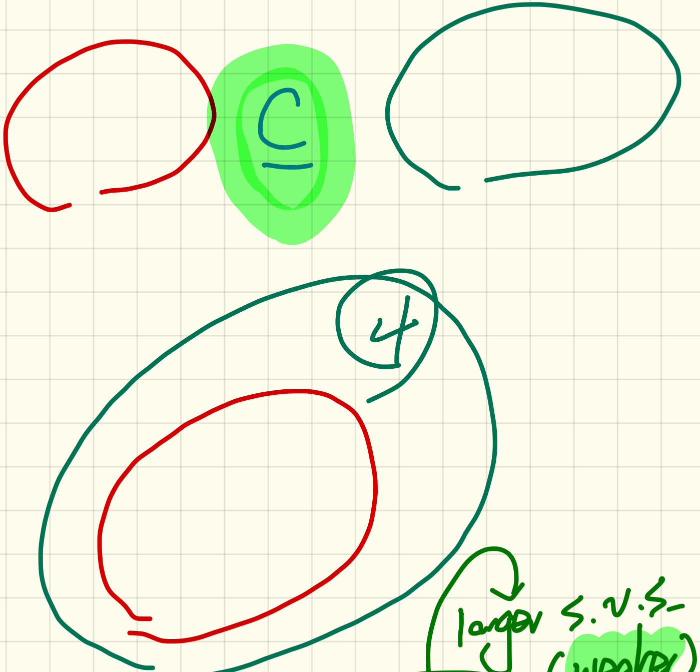
Assertions: Weak vs. Strong

$x > 3$

$4, 5, 6, 7, \ldots, \infty$

$x > 4$

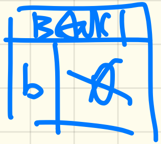
$5, 6, 7, \ldots, \infty$



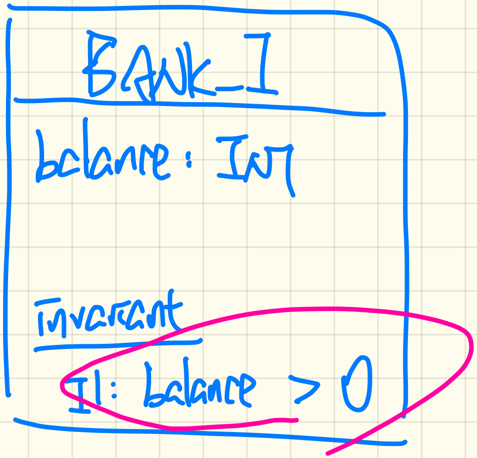
$x > 4 \Rightarrow x > 3$

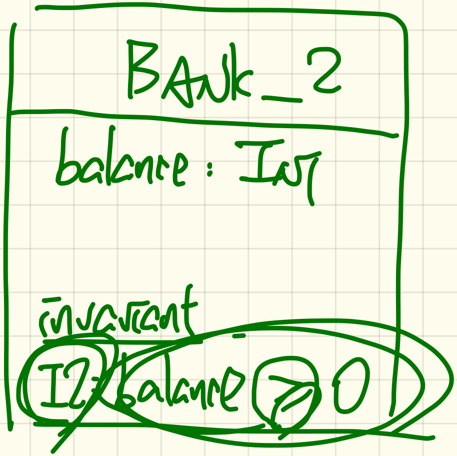
larger s.v.s. (weaker)

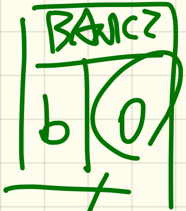
↳ smaller satisfying value set (stronger) runtime



class inv. violation.

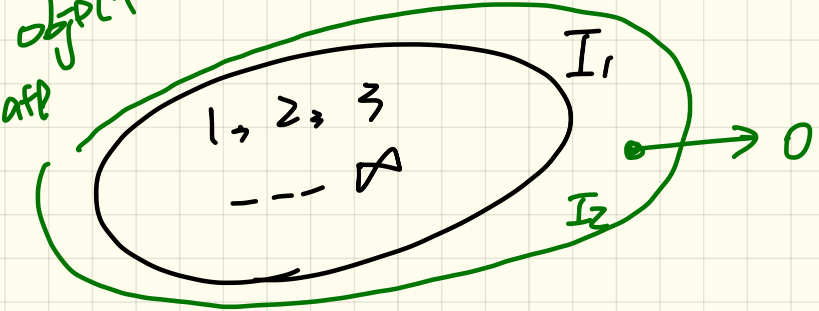






valid object state

Which class invariant is stronger? I1.



$I_1 \Rightarrow I_2$

↓

stronger.

Assertions: Preconditions

```
withdraw_v1(amount: INTEGER)
  require
    P1: amount > 0
```

$0 > 0 \equiv F$

require more

acc.withdraw_v1(0)

precondition violation

$P_1 \Rightarrow P_2$

```
withdraw_v2(amount: INTEGER)
  require
    P2: amount ≥ 0
```

require less

acc.withdraw_v2(0)

no pre. violation

withdraw_v2 is more tolerant on accepting input values

Assertions: Postconditions

```
f1(i: INTEGER): BOOLEAN
  ensure
    Q1: Result = (i > 0) ∨ (i mod 2 = 0)
```

weaker

```
f2(i: INTEGER): BOOLEAN
  ensure
    Q2: Result = (i > 0) ∧ (i mod 2 = 0)
```

✓

stronger → more demanding task for supplier

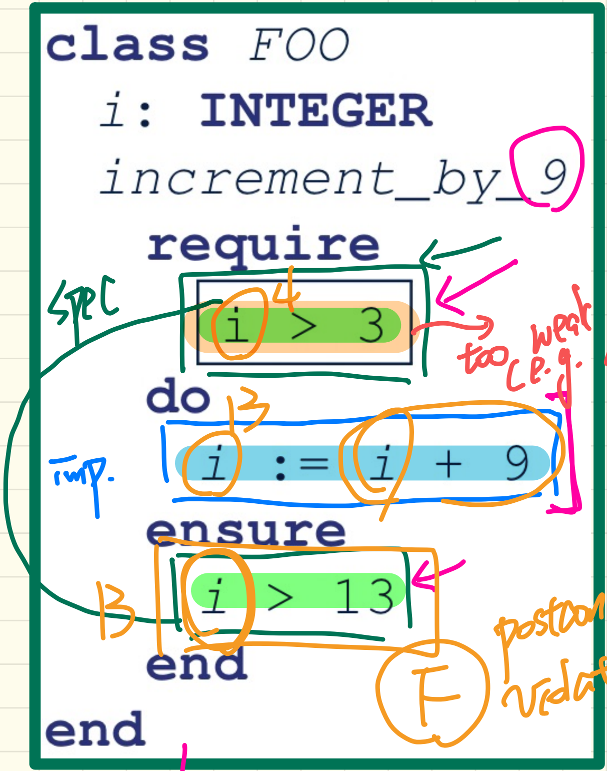
smaller satisfying value set ⇒ more demanding.

3, 5, -2, 7, ..., -4, -6,

Q2

2, 4, 6, 8, 10.

Program Correctness: Example (1)



```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 3
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

↳ not correct.

(relative)

Correctness of program : implementation satisfies specification.

Given valid input (precond. satisf.), executing the implementation will (1) terminate. (2) upon termination, the postcondition is satisf.

4 counter example

4 + 9 > 13 F.

Program Correctness: Example (2)

```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 5
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

5

F

Guiding Principle

cannot be too weak

Correct

i : 6, 7, 8, - - -

$\underbrace{6, 7, 8, \ldots}_{\text{valid input values}}$

$i + 9$ is always > 13

Incorrect (?)

whether a precond. is too strong or not, it's up to the designer.

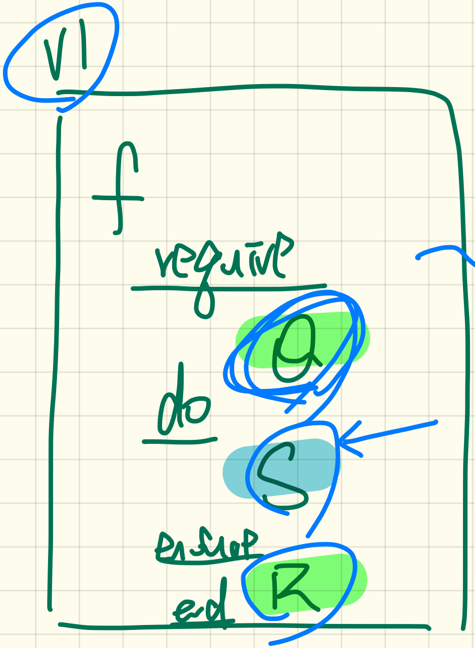
$i > 5$ (precondition) stronger than necessary.

disallow some input value that would not cause postcon. violation!

If 5 was allowed, $5 + 9 = \boxed{14 > 13}$ is

T

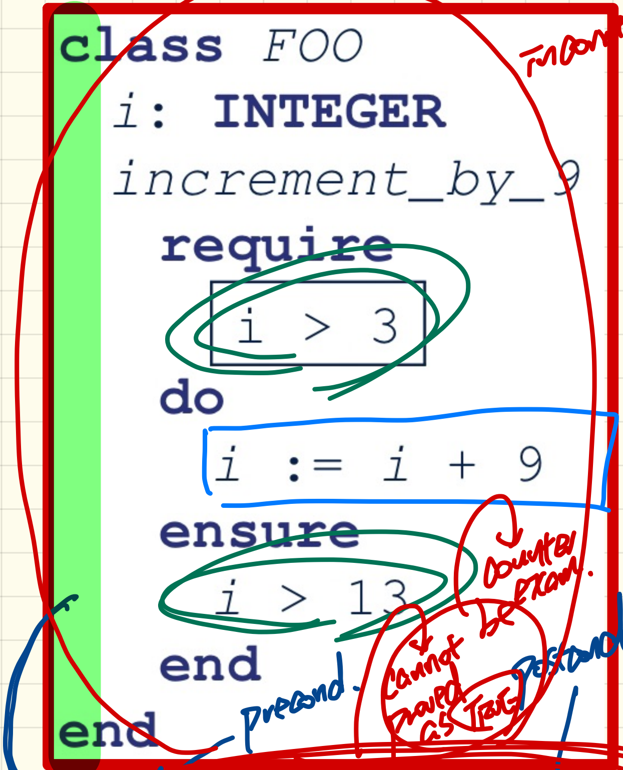
5 currently not considered a valid input.



→ verify whether Q is satisfied, when executing S will establish R.

When you justify that program is incorrect. you may fix: Q, S, R

Hoare Triple → Tony Hoare → Quick sort



```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 3
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

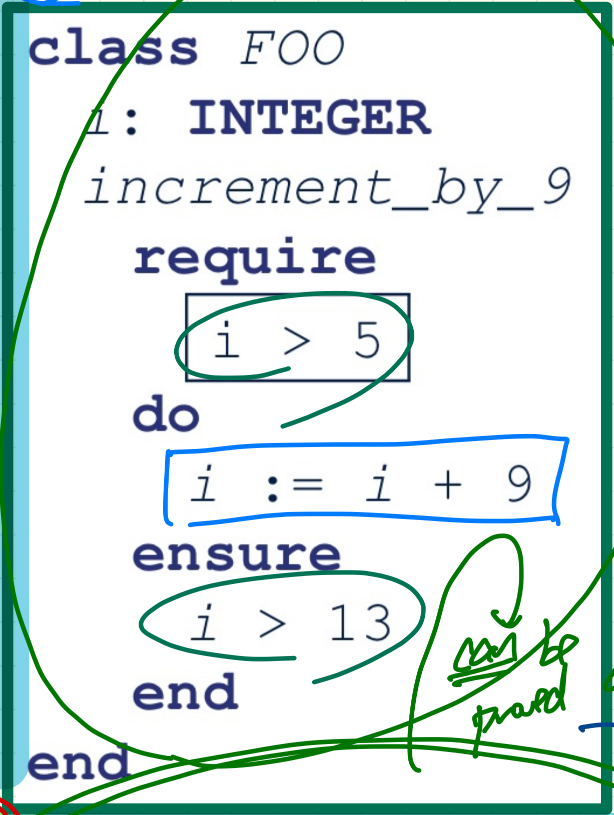
Incorrect

counter example.

cannot be proved as TRUE

precond.

postcond.



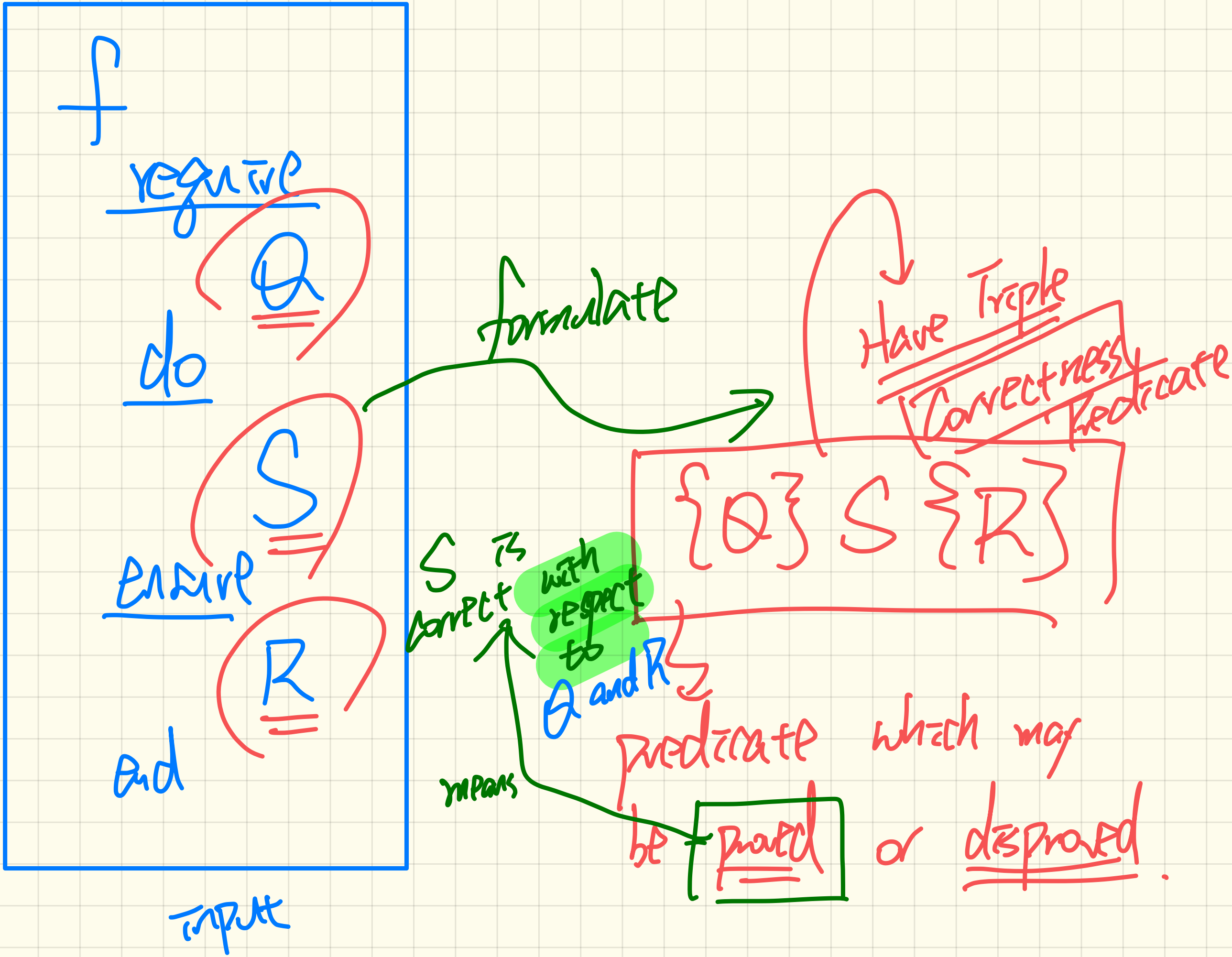
```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 5
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

Correct

can be proved as a theorem

$\{i > 3\}\ i := i + 9\ \{i > 13\}$

$\{i > 5\}\ i := i + 9\ \{i > 13\}$



f

require

Q

do

S

ensure

R

end

input

formulate

Hoare Triple

Correctness Predicate

{Q} S {R}

S is correct with respect to Q and R

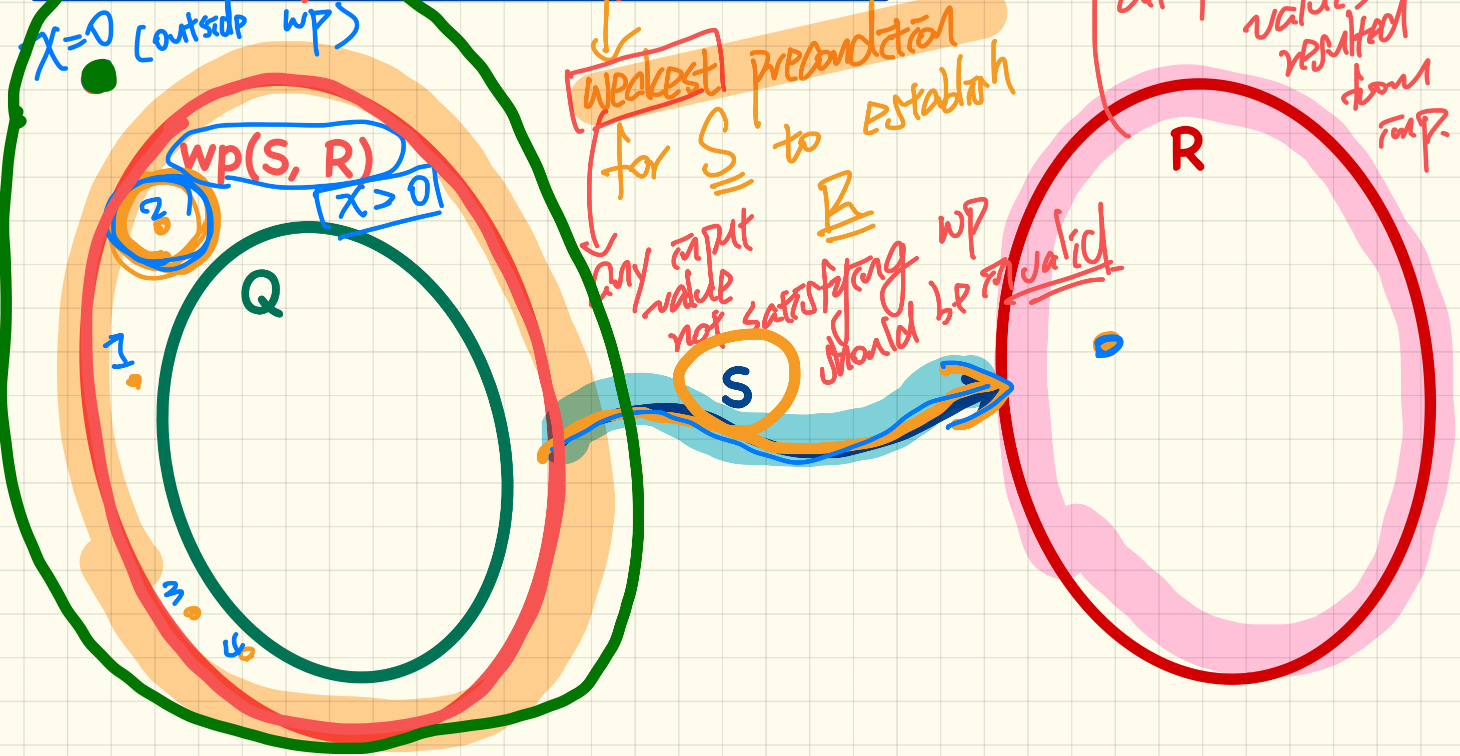
means

predicate which may be proved or disproved.

$x = \underline{old}\ x + 1$

Hoare Triple as a Predicate

$$\{Q\}\ S\ \{R\} \equiv Q \Rightarrow wp(S, R)$$



weakest precondition for S to establish R

any input value not satisfying wp should be invalid

all post-state values resulted from R

Program Correctness: Revisiting Example (1)

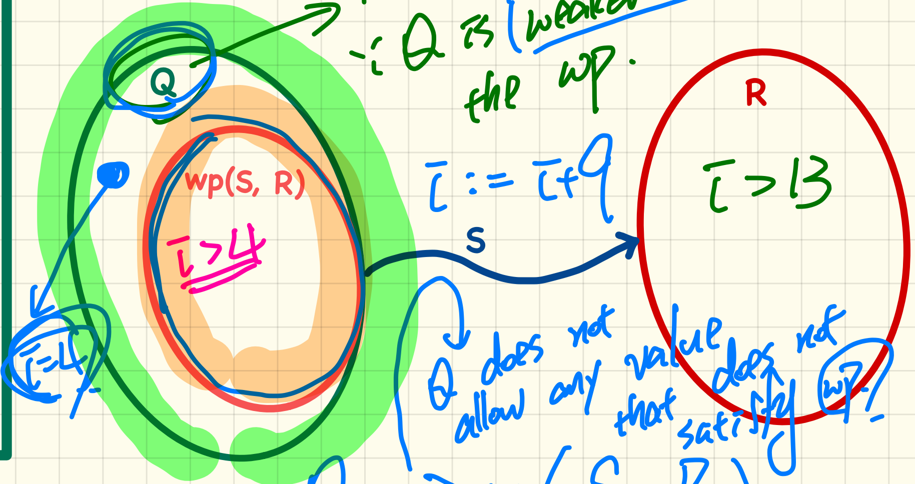
$\boxed{i > 4}$ wp(i := i+9, i > 13)

```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 3
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

incorrect

$$\{Q\}\ S\ \{R\} \equiv Q \Rightarrow \boxed{wp(S, R)}$$

incorrect program ∵ Q is weaker than the wp.



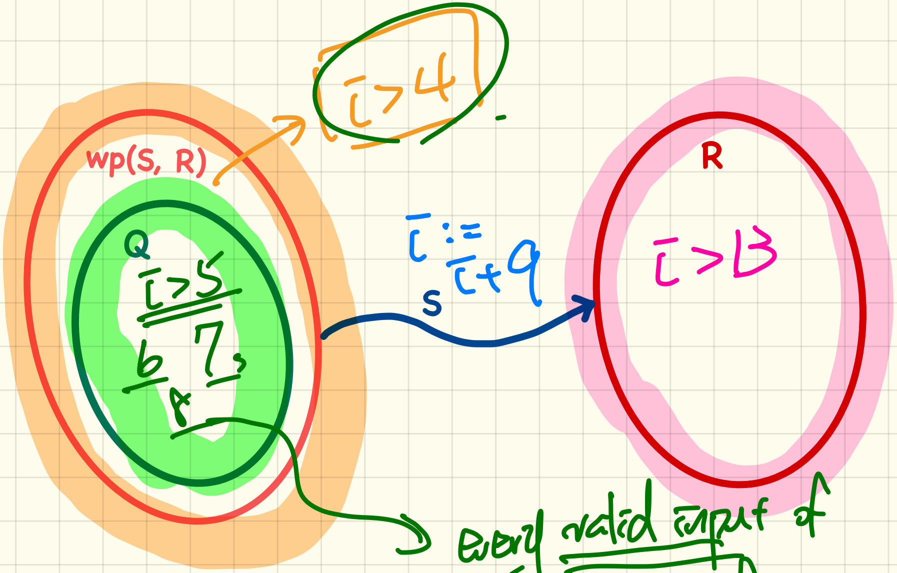
Q does not allow any value that does not satisfy wp?

$Q \Rightarrow wp(S, R)$

Program Correctness: Revisiting Example (2)

```
class FOO
  i: INTEGER
  increment_by_9
    require
      i > 5
    do
      i := i + 9
    ensure
      i > 13
    end
end
```

$$\{Q\}\ S\ \{R\} \equiv Q \Rightarrow wp(S, R)$$



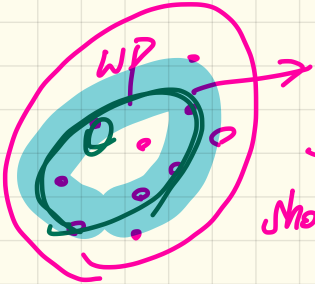
every valid input of your program satisfies wp.

program

```
f
  require
    Q
  do
    S
  ensure
    R
```

↳ program

① formulate into $\{\boxed{Q}\}\ S\ \{R\}$



→ only values satisfying wp should be valid input.

② Caculate $wp(S, R)$

③ Prove or disprove:

$$Q \overset{\subset}{\Rightarrow} wp(S, R)$$

Hoare triple proof (wp)
only makes sure your precond. is not too weak

F

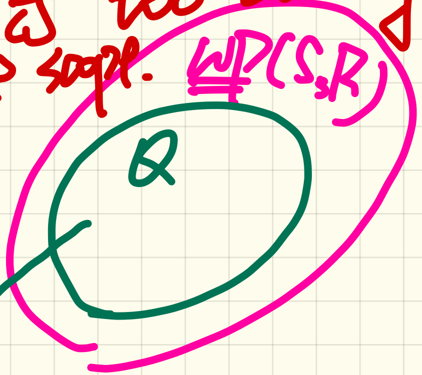
$$\boxed{\underline{\underline{Q}}} \Rightarrow wp(\underline{\underline{S}}, \underline{\underline{R}})$$

whether Q is too strong is beyond the scope.

Q is no weaker than wp.

f require $\boxed{\underline{\underline{false}}}$

by def. $\underline{\underline{correct}}$



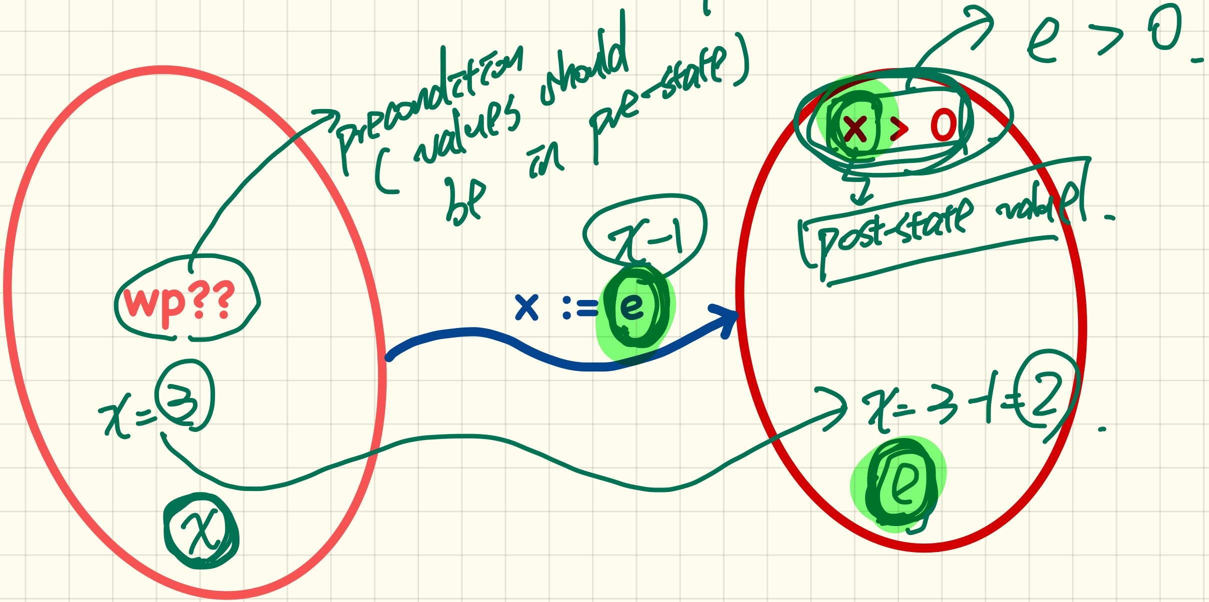
what is the Q that is no weaker than any wp?

Rules of Weakest Precondition: Assignment

$$wp(x := \underset{x-1}{e}, R) = R[x := e]$$

imp: $x := x-1$

post: $x > 0$



Correctness of Programs: Assignment (1)

What is the weakest precondition for a program `x := x + 1` to establish the postcondition $x > x_0$?

$$\{??\}\ \texttt{x := x + 1}\ \{x > x_0\}$$

$(x>0)$ Q True

imp

post.

does it matter?

wp is True

$$Q \Rightarrow \text{True}$$

$$wp(x := x + 1,\ x > x_0)$$

e

pre-state value

$$= \{\ wp \text{ rule for assignment}\ \}$$

$$\overset{x_0+1}{x} > x_0\ [x := x_0 + 1]$$

R

$$= \cancel{x_0} + 1 > \cancel{x_0} = 1 > 0$$

wp.

True

||

For this prog, wp is T, ∴ any precond. is stronger than that.