Recap

Prove program correctness using Hoare Logic

$\{True\}\ \mathcal{P}\ \{z > 8\}$ where program $\mathcal{P}$

```
x = 1
y = x + 1
x = 5
z = x + 5
```

- Sequence rule.
- Assignment rule, backwards with consequence (implicitly).
- Weakest precondition (WP) and strongest postcondition (SP).

Conditionals

$$\frac{\{B \wedge P\}C_1\{Q\}, \{\neg B \wedge P\}C_2\{Q\}}{\{P\}\ if\ B\ C_1\ else\ C_2\{Q\}}$$

- If B is $true$, C_1 is executed;
- If B is $false$ (i.e. $\neg B$), C_2 is executed;
- Both branches should end up with the same post-conditions;

What is the overall precondition?

- $P_1 : \{B \wedge P\}$, push Q up through C_1;
- $P_2 : \{\neg B \wedge P\}$, push Q through C_2;

P is $P_1 \wedge P_2$.

Conditional example

Prove $\{True\}\ \mathcal{P}\ \{z \geq y \land z \geq x\}$ where program $\mathcal{P}$

```
if x > y
    z = x
else
    z = y
```

if branch

$P_1 : \{B \wedge P\}$, push Q up through C_1

```
# {True}
#  {$x > y \vee x \leq y$}
  if x>y
#    {$x > y$}
    z = x
#    {$x > y, z > y, z = x \implies z \geq y \wedge z \geq x$} Assignment, SP
  else
      z = y
```

else branch

$P_2 : \{\neg B \wedge P\}$, push Q through C_2

```
# {True}
#  {x > y ∨ x ≤ y}
  if x>y
    z = x
  else
#    {x ≤ y}
     z = y
#    {x ≤ y, x ≤ z, z = y ⟹ z ≥ y ∧ z ≥ x} Assignment, SP
```

Finished result

Two armed conditional

\# *{True}*

\# $\{x > y \vee x \leq y\}$

`if x>y`

\# $\{x > y\}$

`z = x`

\# $\{x > y, z > y, z = x \implies z \geq y \wedge z \geq x\}$ `Assignment, SP`

`else`

\# $\{x \leq y\}$

`z = y`

\# $\{x \leq y, x \leq z, z = y \implies z \geq y \wedge z \geq x\}$ `Assignment, SP`

\# $\{z \geq x \wedge z \geq y\}$ `Conditional`

Loops

How do we prove correctness of the loop?

```
# n is predefined
  ...
  result = 0
  i = 0
  while i <= n:
      result = result + i
      i = i + 1
```

Loop invariants (I)

A property of a program loop that is true before and after each iteration.

$$\frac{\{C \wedge I\} body \{I\}}{\{I\}\ while(C)\ body\ \{\neg C \wedge I\}}$$

Finding Loop Invariants

```
# n is predefined
  ...
  result = 0
  i = 0
  while i <= n:
      result = result + i
      i = i + 1
```

General strategies for finding loop invariants (I)

1. What is changing in each iteration: i, $result$.
2. Think about a specific iteration: from iteration i(0) to iteration $i+1$(1), only $result$ changed (from 0 to $0+1$), $result = 0 + 1 ... + i - 1$.
3. What do you at the end? $i = n + 1$, the variable result should contain the sum of all nature number from 0 to n.

Finding Loop Invariants

```
# n is predefined
  ...
  result = 0
  i = 0
  while i <= n:
      result = result + i
      i = i + 1
```

Initialization, maintenance, termination

$$I := result = 0 + 1 + ... + i - 1$$

Q & A