Recap

ve pro	ogram correctness using Hoare Logic	
	$\{True\} \ \mathcal{P} \ \{z > 8\}$ where program \mathcal{P}	
1	x = 1	_
2	y = x + 1	
3	x = 5	
4	z = x + 5	

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

Conditionals

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

- 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?

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

P is $P_1 \wedge P_2$.

Conditional example

Prove $\{T\}$	$\{ue\} \ \mathcal{P} \ \{z \geq y \land z \geq x\}$ where program \mathcal{P}	
1	if x > y	
2	z = x	
3	else	
4	z = y	

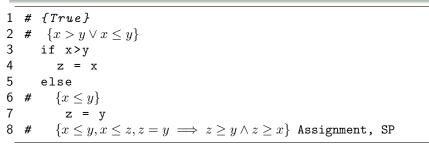
$if \ {\rm branch}$

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

1	#	{True}
2	#	$\{x > y \lor x \le y\}$
3		if x>y
4	#	$\{x > y\}$
5		z = x
6	#	$\{x>y,z>y,z=x\implies z\geq y\wedge z\geq x\}$ Assignment, SP
7		else
8		z = y
-		

else branch

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



Finished result

Two armed conditional

Loops

How do we prove correctness of the loop?

```
1# n is predefined
2 ...
3 result = 0
4 i = 0
5 while i <= n:
6 result = result + i
7 i = i + 1</pre>
```

Loop invariants (I)

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

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

Finding Loop Invariants

```
1# n is predefined
2 ...
3 result = 0
4 i = 0
5 while i <= n:
6 result = result + i
7 i = i + 1
```

General strategies for finding loop invariants (I)

- What is changing in each iteration: i, result.
- 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.
- 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

```
1# n is predefined
2 ...
3 result = 0
4 i = 0
5 while i <= n:
6 result = result + i
7 i = i + 1</pre>
```

Initialization, maintenance, termination

$$I := result = 0 + 1 + \dots + i - 1$$

Q & A