Starter Questions

Feel free to discuss these with your neighbour:

- Consider two states $s_1$ and $s_2$ such that

$$\langle s_1, x := x + 1 \rangle \Rightarrow s_2$$

If predicate $P \equiv (x = y + 1)$ is true for $s_2$ then what does that tell us about $s_1$? How can we change $P$ so that it is true for $s_1$?

- What happens to the value of the expression $x + y$ every time we execute this loop?

```
while x > 0 do (y := y+z ; x := x := x−z)
```
Correctness

I will not re-motivate correctness. We did that previously. The kind of correctness we are concerned with here is a **semantic** notion, not a syntactic one.

As discussed previously, to be able to establish whether a program is correct we need:

- A **specification** of what we want it to do
- A method for checking whether the program meets the specification

Abstractly, a program represents a set of pairs \( \langle s_{\text{start}}, s_{\text{end}} \rangle \) where executing the program on \( s_{\text{start}} \) gives \( s_{\text{end}} \). What does this description miss?

We can **specify** the desired behaviour of a program using predicates on these states.
We introducing the following notion of a specification of program $S$

\[
\{ P \} \ S \{ Q \}
\]

where $P$ is a precondition and $Q$ is a postcondition

The meaning of this is that if $P(s_{\text{start}})$ holds then $Q(s_{\text{end}})$ should hold.

We can define the set of states a predicate denotes i.e.

\[
P_s \equiv \{ s \mid P(s) \}
\]

If $\{ P \} \ S \{ Q \}$ is true then $S$ captures a function from $P_s$ to $Q_s$.

The above notation is a Hoare Triple after Tony Hoare who invented it.
Property: find the maximum of $x$ and $y$ and place it in $z$

Specification: ?
Writing Some Specifications

**Property**: find the maximum of \(x\) and \(y\) and place it in \(z\)

**Specification**: \(\{ \} \ S \{ \ z = \max(x, y) \} \)

Let

\[
\max(x, y) = \begin{cases} 
    x & \text{if } x > 0 \\
    y & \text{otherwise}
\end{cases}
\]

or equivalently

\[
(z = \max(x, y)) \iff (z \geq x \land z \geq y \land (z = x \lor z = y))
\]
**Property**: find the maximum of $x$ and $y$ and place it in $z$

**Specification**: \{ \} $S$ \{ $z = \text{max}(x, y)$ \}

Let $S$ be

$x := 0; y := 0; z := 0$
Property: find the maximum of $x$ and $y$ and place it in $z$

Specification: $\{ x = a \land y = b \} S \{ z = \text{max}(a, b) \}$ where $a, b$ not in $S$

Let $S$ be

$x := 0; y := 0; z := 0$
**Property**: find the maximum of $x$ and $y$ and place it in $z$

**Specification**: \{ $x = a \land y = b$ \} $S$ \{ $z = \text{max}(a, b)$ \} where $a, b$ not in $S$

Let $S$ be

```plaintext
if $x > y$ then $z := x$ else $z := y$
```

We should usually introduce auxiliary variables or argue that we do not need them, how?
Property: Given values in variables $x, y \in \mathbb{N}$ such that $y \neq 0$ set $d$ and $r$ such that $x = d \times y + r$ and $r < y$
Writing Some Specifications

Property: Given values in variables $x, y \in \mathbb{N}$ such that $y \neq 0$ set $d$ and $r$ such that $x = d \times y + r$ and $r < y$

Specification:

$$\{ x \geq 0 \land y > 0 \} \ S \ { x = d \times y + r \land 0 \leq r < y \}$$
A First Informal Argument

Let us consider the program

```
if \ x>y \ then \ z:=x \ else \ z:=y
```

Is it correct with respect to \( \{ \} \ S \ { \ z = \max(x, y) \} \)?

Given any state \( s_{\text{start}} \) what can we say about \( s_{\text{end}} \)?

There are two possible paths through the program:

- If \( x > y \) then we perform \( z := x \), after which \( z = x \land z > y \)
- If \( \neg(x > y) \) then we perform \( z := y \), after which \( z = y \land z \geq x \)

So we know that this holds after the program

\[
(x > y \rightarrow x = z \land z \geq y) \land (y \geq x \rightarrow z = y \land z \geq x)
\]

which is a reasonable definition of \( z = \max(x, y) \)
A Second Informal Argument

Let us now consider the division program

\[
\begin{align*}
r &:= x; \\
d &:= 0; \\
\textbf{while } y \leq r \textbf{ do } (d := d+1; \ r := r-y)
\end{align*}
\]

Is it correct for \( \{ x \geq 0 \land y > 0 \} \ S \{ x = d \times y + r \land 0 \leq r < y \} \)

For every \( s_{start} \) that satisfies \( x \geq 0 \land y > 0 \) do we reach a state \( s_{end} \) that satisfies \( x = d \times y + r \land 0 \leq r < y \)?

Given any state \( s_{start} \) what can we say about \( s_{end} \)?

We cannot say statically how many times the loop will execute!
Recall that

\[ [\text{while } b \text{ do } S] \equiv [\text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}] \]

This means that

\[
\{ P \} \text{ while } b \text{ do } S \{ Q \} \\
\iff \\
\{ P \} \text{ if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip } \{ Q \} \\
\iff \\
\{ P \} \text{ if } b \text{ then } (S; \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}) \text{ else skip } \{ Q \}
\]

The \( Q \) from the end of one iteration of the loop needs to be the \( P \) for the start of the next iteration.
Recall that

\[ \text{while} \, b \, \text{do} \, S \equiv \text{if} \, b \, \text{then} \, (S; \text{while} \, b \, \text{do} \, S) \, \text{else} \, \text{skip} \]

This means that

\[
\{ P \land b \} \; S \; \{ Q \} \\
\iff \\
\{ P \land b \} \; S; \text{while} \; b \; \text{do} \; S \; \{ Q \} \\
\iff \\
\{ P \land b \} \; S; \text{if} \; b \; \text{then} \; (S; \text{while} \; b \; \text{do} \; S) \; \text{else} \; \text{skip} \; \{ Q \} 
\]

The \( Q \) from the end of one iteration of the loop needs to be the \( P \) for the start of the next iteration.
Recall that

\[ \{ \text{while } b \text{ do } S \} \equiv \{ \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip} \} \]

This means that

\[ \{ P \land b \} S \{ Q \} \iff \{ P \land b \} S \{ P \land b \} \text{ while } b \text{ do } S \{ Q \} \]

The \( Q \) from the end of one iteration of the loop needs to be the \( P \) for the start of the next iteration.
Recall that

\[[\text{while } b \text{ do } S] \equiv [\text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}]\]

This means that

\[
\{ P \} \text{ while } b \text{ do } S \{ P \land \lnot b \} \\
\iff \\
\{ P \} \text{ if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip } \{ P \land \lnot b \} \\
\iff \\
\{ P \} \text{ if } b \text{ then } (S; \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}) \text{ else skip} \{ P \land \lnot b \}
\]

\( P \) is an expression that is true on every iteration of the loop; it is a loop invariant.
Let us now consider the division program

\[
\begin{align*}
    r &:= x; \\
    d &:= 0; \\
\textbf{while} \  y \leq r \ \textbf{do} \ (d := d+1; \ r := r-y)
\end{align*}
\]

What is a loop invariant for this loop?

- The variables \(d\) and \(r\) are changed so we need to mention them
- We increase \(d\) by 1 and decrease \(r\) by \(y\)
- So \(d \times y\) increases as much as \(r\) decreases
- So \(K = d \times y + r\) is invariant, but what is \(K\)?
- It needs to hold at the start, where \(r = x\) and \(d = 0\)
- So \(K = 0 \times y + x = x\)
- Making our invariant \(x = (d \times y) + r\), which is unsurprising
Shortly I will introduce a set of axiomatic rules for proving correctness. But we need to separate two kinds of correctness:

- **Partial Correctness.** A program is partially correct if whenever it terminates it satisfies the specification.
- **Total Correctness.** A program is totally correct if it satisfies the specification and always terminates.

We will use 

\[
\{ \ P \ } \ S \ \{ \ Q \ } 
\]

to mean partial correctness and

\[
[ \ P \ ] \ S \ [ \ Q ]
\]

to mean total correctness.
The following statement is true

\[
\{ \} \text{ while } \text{true} \text{ do skip } \{ \text{false} \}
\]
as the program never terminates. This is also true

\[
\{ \} \text{ if } x > y \text{ then } z := x \text{ else (while true do skip) } \{ \text{z = max}(x, y) \}
\]
as whenever it terminates it satisfies the specification! This is also true

\[
[ x \neq x ] \ S [ ]
\]
for any program $S$ as $x \neq x$ will not hold any states.

This shows some of the limitations of this approach
Axiomatic System

\[\text{ass}_p \quad \{ P[x \mapsto A[a]] \} \ x := a \ { P } \]

\[\text{skip}_p \quad \{ P \} \ \text{skip} \ \{ P \} \]

\[\text{comp}_p \quad \{ P \} \ S_1 \ { Q }, \ \{ Q \} \ S_2 \ { R } \]

\[\{ P \} \ S_1; \ S_2 \ { R } \]

\[\text{if}_p \quad \{ P \land B[b] \} \ S_1 \ { Q }, \ \{ P \land \neg B[b] \} \ S_2 \ { Q } \]

\[\{ P \} \ \text{if} \ b \ \text{then} \ S_1 \ \text{else} \ S_2 \ { Q } \]

\[\text{while}_p \quad \{ P \land B[b] \} \ S \ { P } \]

\[\{ P \} \ \text{while} \ b \ \text{do} \ S \ { P \land \neg B[b] } \]
The obvious ones

Skipping doesn’t change anything

\[
\{ P \} \text{ skip } \{ P \}
\]

For sequence \( S_1; S_2 \), the postcondition of \( S_1 \) needs to make the precondition of \( S_2 \) true

\[
\{ P \} S_1 \{ Q \}, \{ Q \} S_2 \{ R \}
\]

\[
\{ P \} S_1; S_2 \{ R \}
\]

For conditionals we can assume the condition and check the statement

\[
\{ P \land B[b] \} S_1 \{ Q \}, \{ P \land \neg B[b] \} S_2 \{ Q \}
\]

\[
\{ P \} \text{ if } b \text{ then } S_1 \text{ else } S_2 \{ Q \}
\]
People often find this rule counterintuituitive

\[
\{ P[x \mapsto A[a]] \} \ x := a \ \{ P \}
\]

as it feels backwards

The handout discusses alternatives and why they don't work

The justification for this approach was given in the starter question. Consider two states \( s_1 \) and \( s_2 \) such that

\[
\langle s_1, x := a \rangle \Rightarrow s_2
\]

We want to find a predicate that is true for \( s_1 \). The states \( s_1 \) and \( s_2 \) only differ in the value of \( x \). If \( P \) is true for \( s_2 \) then we know it is true when \( x \) is replaced by the value for \( a \).
Next Time

- The rest of the rules
- Some detailed examples
- Total correctness