CS 591: Formal Methods in Security and Privacy
Example in Hoare Logic and Non-interference

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu
From the previous classes
Specifications - Hoare triple

Precondition

Program

Postcondition

\[ c : P \implies Q \]

Precondition (a logical formula)

Postcondition (a logical formula)
Validity of Hoare triple

We say that the triple $c : P \Rightarrow Q$ is valid if and only if for every memory $m$ such that $P(m)$ and memory $m'$ such that $\{c\}_m=m'$ we have $Q(m')$.

Is this condition easy to check?
Semantics of Commands

This is defined on the structure of commands:

\[
\begin{align*}
\{\text{abort}\}_m &= \bot \\
\{\text{skip}\}_m &= m \\
\{x := e\}_m &= m[x \leftarrow \{e\}_m] \\
\{c;c'\}_m &= \{c'\}'_m, \quad \text{if} \quad \{c\}_m = m' \\
\{c;c'\}_m &= \bot, \quad \text{if} \quad \{c\}_m = \bot \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \{c_t\}_m, \quad \text{if} \quad \{e\}_m = \text{true} \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \{c_f\}_m, \quad \text{if} \quad \{e\}_m = \text{false} \\
\{\text{while } e \text{ do } c\}_m &= \sup_{n \in \mathbb{N}} \{\text{while}_n e \text{ do } c\}_m \\
\text{where} \\
\text{while}_n e \text{ do } c &= \text{while}_n e \text{ do } c; \text{if } e \text{ then } \text{abort} \text{ else } \text{skip} \\
\end{align*}
\]
Semantics of Commands

This is defined on the structure of commands:

\[
\begin{align*}
\{\text{abort}\}_m &= \bot \\
\{\text{skip}\}_m &= m \\
\{x:=e\}_m &= m[x\leftarrow\{e\}_m] \\
\{c;c'\}_m &= \{c'\}_m' \quad \text{if} \quad \{c\}_m = m' \\
\{c;c'\}_m &= \bot \quad \text{if} \quad \{c\}_m = \bot \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \{c_t\}_m \quad \text{if } \{e\}_m = \text{true} \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \{c_f\}_m \quad \text{if } \{e\}_m = \text{false} \\
\{\text{while } e \text{ do } c\}_m &= \sup_{n\in\mathbb{Nat}} \{\text{while}_n e \text{ do } c\}_m \\
\end{align*}
\]

where

\[
\text{while}_n e \text{ do } c = \text{while}_n e \text{ do } c;\text{if } e \text{ then abort else skip}
\]

and

\[
\text{while}_0 e \text{ do } c = \text{skip} \\
\text{while}_{n+1} e \text{ do } c = \text{if } e \text{ then } (c;\text{while}_n e \text{ do } c) \text{ else skip}
\]
Rules of Hoare Logic:

$\vdash \text{skip}: \ P \Rightarrow P$

$\vdash x := e : \ P[e/x] \Rightarrow P$

$\vdash c ; c' : \ P \Rightarrow Q$

$\vdash c : \ P \Rightarrow R$

$\vdash c' : \ R \Rightarrow Q$

$\vdash c : \ P \Rightarrow Q$

$\vdash c : \ e \land \ P \Rightarrow P$

$\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : \ P \Rightarrow Q$

$\vdash \text{while } e \text{ do } c : \ P \Rightarrow P \land \neg e$

$\vdash c : \ S \Rightarrow R$

$\vdash c : \ R \Rightarrow Q$
Rules of Hoare Logic:

\[ \vdash \text{skip}: P \Rightarrow P \]

\[ \vdash \text{x:=e} : P[e/x] \Rightarrow P \]

\[ \vdash \text{c;c'} : P \Rightarrow Q \]

\[ \vdash \text{if e then c₁ else c₂} : P \Rightarrow Q \]

\[ \vdash \text{while e do c} : P \Rightarrow P \land \neg e \]
Rules of Hoare Logic:

\[\vdash \text{skip}: P \Rightarrow P\]

\[\vdash x := e : P[e/x] \Rightarrow P\]

\[\vdash c; c' : P \Rightarrow Q\]

\[\vdash c : P \Rightarrow R, c' : R \Rightarrow Q \quad \text{implies} \quad c; c' : P \Rightarrow Q\]

\[\vdash c_1 : e \land P \Rightarrow Q, c_2 : \neg e \land P \Rightarrow Q \quad \text{implies} \quad\]

\[\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q\]

\[\vdash c : e \land P \Rightarrow P\]

\[\vdash \text{while } e \text{ do } c : P \Rightarrow P \land \neg e\]
Correctness of a rule

\[ \vdash c_1 : P_1 \Rightarrow Q_1 \quad \ldots \quad \vdash c_n : P_n \Rightarrow Q_n \]

\[ \vdash c : P \Rightarrow Q \]

We say that a rule is correct if given valid triples as described by the assumption(s), we can prove the validity of the triple in the conclusion.
Today 1: More Hoare Logic
Another example

\[
x := 3; \\
y := 1; \\
\text{while } x > 1 \text{ do} \\
\quad y := y + 1; \\
\quad x := x - 1;
\]

\[\vdash \{ \text{true} \} \Rightarrow \{ y = 3 \}\]

What can be a good Invariant?
Another example

\[
\begin{aligned}
x &:= 3; \\
y &:= 1; \\
\text{while } x > 1 \text{ do} & \\
& \quad y := y + 1; \\
& \quad x := x - 1;
\end{aligned}
\]

\[\{\text{true}\} \Rightarrow \{y = 3\}\]

What can be a good Invariant?

\[\text{Inv} = \{y = 4 - x \land x \geq 1\}\]
Another example

\[ \vdash x := 3; y := 1 : \{true\} \Rightarrow \{x = 3 \land 1 = 1 \land y = 4 - x\} \]
Another example
Another example

\[
\begin{align*}
\text{true} & \Rightarrow 3 = 3 & \vdash x := 3 : \{3 = 3\} \Rightarrow \{x = 3\} & \quad x = 3 & \Rightarrow x = 3 \land 1 = 1 & \vdash y := 1 : \{x = 3 \land 1 = 1\} \Rightarrow \{x = 3 \land y = 1\} \\
\vdash x := 3 : \{\text{true}\} \Rightarrow \{x = 3\} & \quad \vdash y := 1 : \{x = 3\} \Rightarrow \{x = 3 \land y = 1\} \\
\vdash x := 3; y := 1 : \{\text{true}\} \Rightarrow \{x = 3 \land y = 1\} & \quad x = 3 \land y = 1 & \Rightarrow x = 3 \land 1 = 1 \land y = 4 - x \\
\vdash x := 3; y := 1 : \{\text{true}\} \Rightarrow \{x = 3 \land 1 = 1 \land y = 4 - x\}
\end{align*}
\]
Another example

\( x = 3 \land y = 1 \land y = 4 - x \Rightarrow y = 4 - x \land x \geq 1 \)

<table>
<thead>
<tr>
<th>while ( x &gt; 1 ) do</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y := y + 1 );</td>
</tr>
<tr>
<td>( x := x - 1 )</td>
</tr>
</tbody>
</table>

: \( \{ y = 4 - x \land x \geq 1 \} \Rightarrow \{ y = 4 - x \land x = 1 \} \quad y = 4 - x \land x = 1 \Rightarrow y = 3 \)

<table>
<thead>
<tr>
<th>while ( x &gt; 1 ) do</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y := y + 1 );</td>
</tr>
<tr>
<td>( x := x - 1 )</td>
</tr>
</tbody>
</table>

: \( \{ x = 3 \land y = 1 \land y = 4 - x \} \Rightarrow \{ y = 3 \} \)
Another example

\[
\begin{align*}
\text{while } x > 1 \
\quad & y := y+1; \\
\quad & x := x-1
\end{align*}
\]

\[
\vdash \{ y = 4 - x \land x \geq 1 \land x > 1 \} \Rightarrow \{ y = 4 - x \land x \geq 1 \}
\]

\[
\begin{align*}
\text{while } x > 1 \
\quad & y := y+1; \\
\quad & x := x-1
\end{align*}
\]

\[
\vdash \{ y = 4 - x \land x \geq 1 \land \neg(x > 1) \} \Rightarrow \{ y = 4 - x \land x = 1 \}
\]

\[
\begin{align*}
\quad & y := y+1; \\
\quad & x := x-1
\end{align*}
\]

\[
x = 3 \land y = 1 \land y = 4 - x \Rightarrow y = 4 - x \land x \geq 1
\]

\[
\begin{align*}
\text{while } x > 1 \\
\quad & y := y+1; \\
\quad & x := x-1
\end{align*}
\]

\[
\vdash \{ y = 4 - x \land x \geq 1 \} \Rightarrow \{ y = 4 - x \land x = 1 \}
\]

\[
y = 4 - x \land x = 1 \Rightarrow y = 3
\]

\[
\begin{align*}
\text{while } x > 1 \\
\quad & y := y+1; \\
\quad & x := x-1
\end{align*}
\]

\[
\vdash \{ x = 3 \land y = 1 \land y = 4 - x \} \Rightarrow \{ y = 3 \}
\]
Another example

\[ y = 4 - x \land x \geq 1 \land x > 1 \Rightarrow y + 1 = 4 - (x - 1) \land x - 1 \geq 1 \]

\[ \vdash \begin{align*}
y &:= y + 1; \\
x &:= x - 1 \end{align*} : \{ y + 1 = 4 - (x - 1) \land x - 1 \geq 1 \} \Rightarrow \{ y = 4 - x \land x \geq 1 \}\]

\[ \vdash \begin{align*}
y &:= y + 1; \\
x &:= x - 1 \end{align*} : \{ y = 4 - x \land x \geq 1 \land x > 1 \} \Rightarrow \{ y = 4 - x \land x \geq 1 \}\]

while \( x > 1 \) do: \{ y = 4 - x \land x \geq 1 \} \Rightarrow \{ y = 4 - x \land x \geq 1 \land \neg(x > 1) \}

\[ \vdash \begin{align*}
y &:= y + 1; \\
x &:= x - 1 \end{align*} \Rightarrow \{ y = 4 - x \land x \geq 1 \land \neg(x > 1) \} \Rightarrow \{ y = 4 - x \land x = 1 \}\]

\[ x = 3 \land y = 1 \land y = 4 - x \Rightarrow y = 4 - x \land x \geq 1 \]

while \( x > 1 \) do

\[ \vdash \begin{align*}
y &:= y + 1; \\
x &:= x - 1 \end{align*} : \{ y = 4 - x \land x \geq 1 \} \Rightarrow \{ y = 4 - x \land x = 1 \} \quad y = 4 - x \land x = 1 \Rightarrow y = 3\]

while \( x > 1 \) do

\[ \vdash \begin{align*}
y &:= y + 1; \\
x &:= x - 1 \end{align*} : \{ x = 3 \land y = 1 \land y = 4 - x \} \Rightarrow \{ y = 3 \} \]
Another example

\(\vdash y := y+1; \{y + 1 = 4 - (x - 1) \land x - 1 \geq 1\} \Rightarrow \{y = 4 - x \land x \geq 1\}\)

\(\vdash x := x-1; \{y = 4 - (x - 1) \land x - 1 \geq 1\} \Rightarrow \{y = 4 - x \land x \geq 1\}\)

\(y = 4 - x \land x \geq 1 \land x > 1 \Rightarrow y + 1 = 4 - (x - 1) \land x - 1 \geq 1\)

\(\vdash y := y+1; \\{y + 1 = 4 - (x - 1) \land x - 1 \geq 1\} \Rightarrow \{y = 4 - x \land x \geq 1\}\)

\(\vdash x := x - 1; \\{y = 4 - x \land x \geq 1 \land x > 1\} \Rightarrow \{y = 4 - x \land x \geq 1\}\)

while \(x > 1\) do: \(\{y = 4 - x \land x \geq 1\} \Rightarrow \{y = 4 - x \land x \geq 1 \land \neg(x > 1)\}\)

\(\vdash y := y+1; \quad x := x-1 \quad \{y = 4 - x \land x \geq 1 \land \neg(x > 1)\} \Rightarrow \{y = 4 - x \land x = 1\}\)

\(x = 3 \land y = 1 \land y = 4 - x \Rightarrow y = 4 - x \land x \geq 1\)

while \(x > 1\) do

\(\vdash y := y+1; \quad x := x-1 \quad \{y = 4 - x \land x \geq 1\} \Rightarrow \{y = 4 - x \land x = 1\} \quad y = 4 - x \land x = 1 \Rightarrow y = 3\)

while \(x > 1\) do

\(\vdash y := y+1; \quad x := x-1 \quad \{x = 3 \land y = 1 \land y = 4 - x\} \Rightarrow \{y = 3\}\)
Another example

\[
\begin{align*}
x & := 3; & \{\text{true}\} \Rightarrow \{x = 3 \land 1 = 1 \land y = 4 - x\} & \Downarrow x := x - 1; & \{x = 3 \land y = 1 \land y = 4 - x\} \Rightarrow \{y = 3\} \\
y & := 1; & \\
x := 3; \quad y := 1; \quad \text{while } x > 1 \text{ do} & : \{\text{true}\} \Rightarrow \{y = 3\}
\end{align*}
\]
What happens if the loop does not end?
Another example

\(\vdash \text{while true do skip} : \{true\} \Rightarrow \{false\}\)

Can we prove it?
Another example

\[
\begin{align*}
\text{while true do skip} & : \{true\} \Rightarrow \{false\} \\
\text{Can we prove it?} \\
\text{What can be a good Invariant?}
\end{align*}
\]
Another example

\[\text{while true do skip} : \{true\} \Rightarrow \{false\}\]

Can we prove it?

What can be a good Invariant?

\[\text{Inv} = \{true\}\]
Another example

\[\vdash \text{while true do skip} : \{true\} \Rightarrow \{false\}\]

\[\vdash c : \text{true} \land \text{true} \Rightarrow \text{true}\]

\[\vdash \text{while true do skip} : \text{true} \Rightarrow \text{true} \land \neg \text{true}\]
Partial vs Total correctness

Partial correctness: the definition of validity requires the postcondition to hold only if the program terminates.

Total correctness: the definition of validity requires the program to terminate and the postcondition to hold.
Total Correctness
Validity of Hoare triple

We say that the triple \( c : P \Rightarrow Q \) is valid if and only if for every memory \( m \) such that \( P(m) \) there exists a memory \( m' \) such that \( \{c\}_m = m' \) and \( Q(m') \).
Total Correctness

Hoare Logic

• All the rules except the ones for abort and while support total correctness.
• We could give a total correctness rule for while.
How do we know that these are the right rules?
Soundness

If we can derive $\vdash c : P \Rightarrow Q$ through the rules of the logic, then the triple $c : P \Rightarrow Q$ is valid.
Are the rules we presented sound?
Completeness

If a triple $c : P \Rightarrow Q$ is valid, then we can derive $\vdash c : P \Rightarrow Q$ through the rules of the logic.
Are the rules we presented complete?
Relative Completeness

\[
\begin{array}{c}
P \Rightarrow S \\
\vdash \text{c : } S \Rightarrow R \\
R \Rightarrow Q
\end{array}
\]

\[
\vdash \text{c : } P \Rightarrow Q
\]
Relative Completeness

\[ P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q \]

\[ \vdash c : P \Rightarrow Q \]

If a triple \( c : \text{Pre} \Rightarrow \text{Post} \) is valid, and we have an oracle to derive all the true statements of the form \( P \Rightarrow S \) and of the form \( R \Rightarrow Q \), which we can use in applications of the conseq rule, then we can derive \( \vdash c : \text{Pre} \Rightarrow \text{Post} \) through the rules of the logic.
Today 2: weakest precondition calculus
Predicate Transformer Semantics

Given a program \( c \) and an assertion \( P \), we can define an assertion \( \text{wp}(c, P) \) which is the weakest precondition of \( c \) and \( P \), i.e. \( c : \text{wp}(c, P) \Rightarrow P \) is a valid triple, and for every triple \( c : Q \Rightarrow P \) we have \( Q \Rightarrow \text{wp}(c, P) \).
Weakest precondition

This is defined on the structure of commands:

\[
\begin{align*}
\wp(\text{abort}, P) &= \text{false} \\
\wp(\text{skip}, P) &= P \\
\wp(x := e, P) &= P[x \leftarrow \{e\}_m] \\
\wp(c; c', P) &= \wp(c, \wp(c', P)) \\
\wp(\text{if } e \text{ then } c_\text{t} \text{ else } c_\text{f}, P) &= (e \Rightarrow \wp(c_\text{t}, P)) \land (\neg e \Rightarrow \wp(c_\text{t}, P)) \\
\wp(\text{while } e \text{ do } c, P) &= \exists n \in \text{Nat} \ P_n \text{ where}
\end{align*}
\]
Weakest precondition

This is defined on the structure of commands:

\[
\begin{align*}
wp(\text{abort}, P) &= \text{false} \\
wp(\text{skip}, P) &= P \\
wp(x := e, P) &= P[x \leftarrow \{e\}_m] \\
wp(c;c', P) &= wp(c, wp(c', P)) \\
wp(\text{if } e \text{ then } c_t \text{ else } c_f, P) &= (e \Rightarrow wp(c_t, P)) \land (\neg e \Rightarrow wp(c_t, P)) \\
wp(\text{while } e \text{ do } c, P) &= \exists n \in \mathbb{Nat} \ P_n \text{ where} \\
&\quad P_0 = \neg e \land P \\
&\quad P_{n+1} = e \land wp(c, P_n)
\end{align*}
\]
Today 3: security as information flow control
Some Examples of Security Properties

• Access Control
• Encryption
• Malicious Behavior Detection
• Information Filtering
• Information Flow Control
Some Examples of Security Properties

• Access Control
• Encryption
• Malicious Behavior Detection
• Information Filtering
• Information Flow Control
Private vs Public

We want to distinguish confidential information that need to be kept secret from nonconfidential information that can be accessed by everyone.

We assume that every variable is tagged with one either public or private.

\[
\begin{align*}
  x & : \text{public} \\
  x & : \text{private}
\end{align*}
\]
Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.
Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.
Is this program secure?

\[
x : \text{private} \\
y : \text{public} \\
x := y
\]
Is this program secure?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
x & := y
\end{align*}
\]

Secure
Is this program secure?

\[
x: \text{private} \\
y: \text{public} \\
y := x
\]
Is this program secure?

\[
x: \text{private} \\
y: \text{public}
\]

\[y := x\]

Insecure
Is this program secure?

\[
x: \text{private} \\
y: \text{public} \\
y := x; \\
y := 5
\]
Is this program secure?

```
x: private
y: public

y := x;
y := 5
```

Secure
Is this program secure?

x: private
y: public

if y mod 3 = 0 then
    x := 1
else
    x := 0
Is this program secure?

\[ \text{x: private} \]
\[ \text{y: public} \]

\[ \text{if } y \mod 3 = 0 \text{ then } \]
\[ \text{x := 1} \]
\[ \text{else} \]
\[ \text{x := 0} \]
Is this program secure?

x:private
y:public

if x mod 3 = 0 then
  y:=1
else
  y:=0
Is this program secure?

\[
x: \text{private} \\
y: \text{public}
\]

\[
\text{if } x \mod 3 = 0 \text{ then} \\
y := 1 \\
\text{else} \\
y := 0
\]

Insecure
How can we formulate a policy that forbids flows from private to public?
Low equivalence

Two memories $m_1$ and $m_2$ are low equivalent if and only if they coincide in the value that they assign to public variables.

In symbols: $m_1 \sim_{\text{low}} m_2$
Noninterference

A program $\text{prog}$ is noninterferent if and only if, whenever we run it on two memories $m_1$ and $m_2$ that are low equivalent, we obtain two memories $m_1'$ and $m_2'$ which are also low equivalent.
Noninterference

In symbols

\( m_1 \sim_{\text{low}} m_2 \) and \( \{c\}_{m_1} = m_1' \) and \( m_2' \{c\}_{m_2} = m_2' \)

implies \( m_1' \sim_{\text{low}} m_2' \)
Does this program satisfy noninterference?

\[
\begin{array}{|c|}
\hline
x: \text{private} \\
y: \text{public} \\
x := y \\
\hline
\end{array}
\]
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
x := y
\]

Yes
Does this program satisfy noninterference?

\[ x : \text{private} \]
\[ y : \text{public} \]
\[ x := y \]

Yes

\[ m_{\min_1} = [x=n_1, y=k] \]
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
x := y
\]

\[
m^{\text{in}_1} = [x=n_1, y=k] \\
m^{\text{in}_2} = [x=n_2, y=k]
\]
Does this program satisfy noninterference?

\[ x : \text{private} \]
\[ y : \text{public} \]
\[ x := y \]

\[ m_{\text{in}_1} = [x = n_1, y = k] \]
\[ m_{\text{in}_2} = [x = n_2, y = k] \]

\[ m_{\text{out}_1} = [x = k, y = k] \]
\[ m_{\text{out}_2} = [x = k, y = k] \]

Yes
Does this program satisfy noninterference?

\[
\begin{array}{c}
x : \text{private} \\
y : \text{public} \\
y := x
\end{array}
\]
Does this program satisfy noninterference?

\[
x : \text{private} \\
y : \text{public} \\
y := x
\]

No
Does this program satisfy noninterference?

\[
\begin{array}{l}
x: \text{private} \\
y: \text{public} \\
y := x
\end{array}
\]

\[m_{\text{in}_1} = [x = n_1, y = k]\]

No
Does this program satisfy noninterference?

\[
\begin{array}{l}
x : \text{private} \\
y : \text{public} \\
y := x
\end{array}
\]

\[
m_{\text{in}_1} = [x = n_1, y = k] \\
m_{\text{in}_2} = [x = n_2, y = k]
\]

No
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
y := x
\]

\[
\begin{align*}
  m_{\text{in}\,1} &= [x=n_1, y=k] \\
  m_{\text{out}\,1} &= [x=n_1, y=n_1] \\
  m_{\text{in}\,2} &= [x=n_2, y=k] \\
  m_{\text{out}\,2} &= [x=n_2, y=n_2]
\end{align*}
\]

No
Does this program satisfy noninterference?

\[
\begin{align*}
&x: \text{private} \\
&y: \text{public} \\
&y := x \\
&y := 5
\end{align*}
\]
Does this program satisfy noninterference?

```plaintext
x:private
y:public
y:=x
y:=5
```

Yes
Does this program satisfy noninterference?

Yes

\[
\begin{array}{c}
x: \text{private} \\
y: \text{public}
\end{array}
\]

\[
y := x \\
y := 5
\]

\[m_{\text{in}_1} = [x = n_1, y = k]\]
Does this program satisfy noninterference?

```
x: private
y: public

y := x
y := 5
```

\( \text{min}_1 = [x = n_1, y = k] \quad \text{min}_2 = [x = n_2, y = k] \)

Yes
Does this program satisfy noninterference?

x: private
y: public

\[ y := x \]
\[ y := 5 \]

\[ m_{\text{in}}^1 = [x = n_1, y = k] \]
\[ m_{\text{out}}^1 = [x = n_1, y = 5] \]

\[ m_{\text{in}}^2 = [x = n_2, y = k] \]
\[ m_{\text{out}}^2 = [x = n_2, y = 5] \]

Yes
Does this program satisfy noninterference?

\begin{verbatim}
x:private
y:public
if \ y \ mod \ 3 = 0 \ then
  x:=1
else
  x:=0
\end{verbatim}
Does this program satisfy noninterference?

x:private
y:public
if y mod 3 = 0 then
  x:=1
else
  x:=0

Yes
Does this program satisfy noninterference?

```
x: private
y: public
if y mod 3 = 0 then
  x := 1
else
  x := 0
```

\[ m_{\min_1} = \{x=n_1, y=6\} \]
Does this program satisfy noninterference?

\[
\begin{array}{l}
\text{x: private} \\
\text{y: public} \\
\text{if } y \mod 3 = 0 \text{ then} \\
\quad x := 1 \\
\text{else} \\
\quad x := 0
\end{array}
\]

\[m_{\text{in}_1} = [x = n_1, y = 6]\]
\[m_{\text{in}_2} = [x = n_2, y = 6]\]

Yes
Does this program satisfy noninterference?

\[\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} & \\
\quad x &= 1 \\
\text{else} & \\
\quad x &= 0
\end{align*}\]

Yes

\[\begin{align*}
m_{\text{in}1} &= [x=n_1, y=6] & m_{\text{in}2} &= [x=n_2, y=6] \\
m_{\text{out}1} &= [x=1, y=6] & m_{\text{out}2} &= [x=1, y=6]
\end{align*}\]
Does this program satisfy noninterference?

x: private
y: public
if x mod 3 = 0 then
  y := 1
else
  y := 0
Does this program satisfy noninterference?

\[
x : \text{private} \\
y : \text{public} \\
\text{if } x \mod 3 = 0 \text{ then} \\
\quad y := 1 \\
\text{else} \\
\quad y := 0
\]

No
Does this program satisfy noninterference?

\[
\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
\text{if} \ x \ \text{mod} \ 3 &= 0 \ \text{then} \\
\quad y &= 1 \\
\text{else} \\
\quad y &= 0
\end{align*}
\]

\[m^1_{in} = [x=6, y=k]\]
Does this program satisfy noninterference?

x: private  
y: public  
if \( x \mod 3 = 0 \) then
  y := 1
else
  y := 0

\( m^{\text{in}_1} = [x=6, y=k] \)

\( m^{\text{in}_2} = [x=5, y=k] \)

No
Does this program satisfy noninterference?

x: private
y: public
if \( x \mod 3 = 0 \) then
  y := 1
else
  y := 0

\( m^{\text{in}}_1 = [x=6, y=k] \)
\( m^{\text{out}}_1 = [x=6, y=1] \)

\( m^{\text{in}}_2 = [x=5, y=k] \)
\( m^{\text{out}}_2 = [x=5, y=0] \)

No
Does this program satisfy noninterference?

s1: public
s2: private
r: private
i: public

proc Compare (s1: list[n] bool, s2: list[n] bool)
i := 0;
r := 0;
while i < n \ r=0 do
  if not (s1[i] = s2[i]) then
    r := 1
  i := i + 1
Does this program satisfy noninterference?

s1: public
s2: private
r: private
i: public

proc Compare (s1: list[n] bool, s2: list[n] bool)
i := 0;
r := 0;
while i < n \( \land \) r = 0 do
    if not (s1[i] = s2[i]) then
        r := 1
    i := i + 1

No
How can we prove our programs noninterferent?