Lecture 3 – Reasoning about Loops
Quick Recap (5 min)

Full Correctness Toolkit
Correctness Toolkit

- Forward and backward reasoning for...
  - assignments
  - if statements
  - loops
  - (essentially) all code can be rewritten to use just these

- Forward / backward reasoning fill in post-/pre-condition

- Check places where assertions meet each other
  - top assertion must imply the bottom assertion
Consider a while-loop (other loop forms not too different) with a loop invariant \( I \).

\[
\begin{align*}
\{ \{ P \} \} & \quad S_1 \\
\{ \{ \text{Inv: } I \} \} & \quad \text{while} \ (\text{cond}) \\
\quad & \quad S_2 \\
\quad & \quad S_3 \\
\{ \{ Q \} \} &
\end{align*}
\]

Informally, we need:
- \( I \) holds initially
- \( I \) holds each time around
- \( Q \) holds after we exit

Formally, we need validity of:
- \( \{ \{ P \} \} \ S_1 \ \{ \{ I \} \} \)
- \( \{ \{ I \text{ and cond} \} \} \ S_2 \ \{ \{ I \} \} \)
- \( \{ \{ I \text{ and not cond} \} \} \ S_3 \ \{ \{ Q \} \} \)
Q & A
Another Example
Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
&\{ \ 0 \leq n \leq b.\text{length} \ \} \\
i &= k = 0; \\
\text{while} \ (i \neq n) \ {\{} \\
&\quad \text{if} \ (b[i] < 0) \ {\{} \\
&\quad\quad \text{swap} \ b[i], \ b[k]; \\
&\quad\quad k = k + 1; \\
&\quad {\}} \\
&\quad i = i + 1; \\
{\}}
\end{align*}
\]

\(
\{ \ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[n-1] \ \}
\)

(Also: precondition is true throughout the code. I’ll skip writing that to save space...)  
(Also: \( b \) contains the same numbers since we use swaps.)
Example: partition array

Consider the following code to put the negative values at the beginning of array $b$:

\[
\begin{align*}
&\{\ 0 \leq n \leq b\.length \}\} \\
i = k = 0; \\
&\{\ Inv: b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \}\} \\
while \ (i \neq n) \{ \\
&\quad if \ (b[i] < 0) \{ \\
&\quad\quad swap \ b[i], \ b[k]; \\
&\quad\quad k = k + 1; \\
&\quad\} \\
&\quad i = i + 1; \\
&\} \\
&\{\ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[n-1] \}\}
\end{align*}
\]
Consider the following code to put the negative values at the beginning of array $b$:

\[
\begin{align*}
\text{\{0} \leq n \leq \text{b.length}\} \\
i = k = 0; \\
\text{\{Inv: b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1]\}} \\
\text{while (i} \neq \text{n)} \{ \\
\quad \text{if (b[i] < 0)} \{ \\
\quad\quad \text{swap b[i], b[k];} \\
\quad\quad k = k + 1; \\
\quad\} \\
\quad i = i + 1; \\
\} \\
\text{\{b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[n-1]\}}
\end{align*}
\]

- $I$ holds initially:
  - $b[0], \ldots, b[-1]$ is empty
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
\{ \{ \ 0 \leq n \leq b.\text{length} \} \} \\
i &= k = 0; \\
\{ \{ \text{Inv: } b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \} \} \\
\text{while} \ (i \neq n) \ {\{} \\
&\quad \text{if} \ (b[i] < 0) \ {\{} \\
&\quad\quad \text{swap } b[i], \ b[k]; \\
&\quad\quad \ k = k + 1; \\
&\quad {\}} \\
&\quad i = i + 1; \\
{\}} \\
\{ \{ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[n-1] \} \}
\end{align*}
\]

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
Example: partition array

Consider the following code to put the negative values at the beginning of array $b$:

```java
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
  if (b[i] < 0) {
    swap b[i], b[k];
    k = k + 1;
  }
  i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- $I$ holds initially
- $I$ and $i = n$ implies postcondition
- $I$ is maintained by loop body?
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
\{ \ & 0 \leq n \leq b.length \} \\
& i = k = 0; \\
\{ \ & \text{Inv: } b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \} \\
& \text{while (} i \neq n \text{) } \{ \\
& \quad \text{if (} b[i] \ < \ 0 \text{) } \{ \\
& \quad \quad \text{swap } b[i], b[k]; \\
& \quad \quad k = k + 1; \\
& \quad \} \\
& \quad i = i + 1; \\
& \quad \} \\
\{ \ & b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[n-1] \}
\end{align*}
\]

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
- \( I \) is maintained by loop body?
Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
\text{\{0} \leq n \leq b.length \}\} \\
i = k = 0; \\
\text{\{Inv: b[0], ..., b[k-1] < 0 \leq b[k], ..., b[i-1] \}\}} \\
\text{while (i != n) { \\
\text{if (b[i] < 0) { \\
\quad \text{swap b[i], b[k];} \\
\quad k = k + 1; \\
\text{}} } \\
\text{{ b[0], ..., b[k-1] < 0 \leq b[k], ..., b[i] \}} \\
\text{i = i + 1; } \\
\text{}} \\
\text{\{ b[0], ..., b[k-1] < 0 \leq b[k], ..., b[n-1] \}}
\end{align*}
\]

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
- \( I \) is maintained by loop body?
Example: partition array

Consider the following code to put the negative values at the beginning of array $b$:

```java
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
    if (b[i] < 0) {
        swap b[i], b[k];
        k = k + 1;
    }
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- $I$ holds initially
- $I$ and $i = n$ implies postcondition
- $I$ is maintained by loop body?
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

```java
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
    if (b[i] < 0) {
        swap b[i], b[k];
        k = k + 1;
    } else {
    }
    {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
    i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
- \( I \) is maintained by loop body?
Example: partition array

Consider the following code to put the negative values at the beginning of array b:

```java
{{ 0 <= n <= b.length }}

i = k = 0;

{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}

while (i != n) {
    if (b[i] < 0) {
        {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
        swap b[i], b[k];
        k = k + 1;
    } else {
        {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
    }
    i = i + 1;
}
```

- I holds initially
- I and i = n implies postcondition
- I is maintained by loop body?
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
&\{ 0 \leq n \leq b.\text{length} \}\} \\
&i = k = 0; \\
&\{\text{Inv: } b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \}\} \\
&\text{while } (i \neq n) \{ \\
&\quad \text{if } (b[i] < 0) \{ \\
&\quad\quad \{\text{b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] and b[i] < 0}\}\} \\
&\quad\quad \text{swap } b[i], b[k]; \\
&\quad\quad k = k + 1; \\
&\quad\quad \{\text{b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i]} \}\} \\
&\quad \text{else } \{ \\
&\quad\quad \{\text{b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] and b[i] \geq 0}\}\} \\
&\quad\quad \{\text{b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i]} \}\} \\
&\quad \} \\
&i = i + 1; \\
&\}
\end{align*}
\]

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
- \( I \) is maintained by loop body?

\[\text{equivalent}\]
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

```java
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
    if (b[i] < 0) {
        {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] and b[i] < 0 }}
        swap b[i], b[k];
        k = k + 1;
        {{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[i] }}
    }
    i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
- \( I \) is maintained by loop body?

Remain to check this…
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
\{ & \ 0 \leq n \leq b.\text{length} \} \\
i = k = 0; \\
\{ & \ \text{Inv: } b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \} \\
\text{while} \ (i \neq n) \ {\{} \\
& \quad \text{if} \ (b[i] < 0) \ {\{} \\
& \quad \quad \{ & \ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \text{ and } b[i] < 0 \} \\
& \quad \quad \text{swap } b[i], \ b[k]; \\
& \quad \quad \ k = k + 1; \\
& \quad \quad \{ & \ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i] \} \\
& \quad {\}} \\
& \quad i = i + 1; \\
& {\}} \\
\{ & \ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[n-1] \}
\end{align*}
\]

- \( \text{I} \) holds initially
- \( \text{I} \) and \( i = n \) implies postcondition
- \( \text{I} \) is maintained by loop body?
Example: partition array

Consider the following code to put the negative values at the beginning of array \( b \):

\[
\begin{align*}
\texttt{\{0} \leq n \leq b.length \} \\
i = k = 0; \\
\texttt{\{Inv: b[0], ..., b[k-1] < 0 \leq b[k], ..., b[i-1] \}} \\
\text{while (i != n) \{} \\
\quad \text{if (b[i] < 0) \{} \\
\quad \quad \texttt{\{b[0], ..., b[k-1] < 0 \leq b[k], ..., b[i-1]} \text{ and } b[i] < 0 \} \\
\quad \quad \text{swap b[i], b[k];} \\
\quad \quad \texttt{\{b[0], ..., b[k] < 0 \leq b[k+1], ..., b[i]\}} \\
\quad \quad k = k + 1; \\
\quad \} \\
\quad i = i + 1; \\
\} \\
\texttt{\{b[0], ..., b[k-1] < 0 \leq b[k], ..., b[n-1]\}}
\end{align*}
\]

- \( I \) holds initially
- \( I \) and \( i = n \) implies postcondition
- \( I \) is maintained by loop body?

This is a valid triple. (Takes some thought.)
Example: partition array

\[
\{ \{ b[0], \ldots, b[k-1] < 0 \leq b[k], \ldots, b[i-1] \text{ and } b[i] < 0 \} \}
\]

\[
\text{swap } b[i], b[k];
\]

\[
\{ \{ b[0], \ldots, b[k] < 0 \leq b[k+1], \ldots, b[i] \} \}
\]

yellow is < 0
purple is >= 0
Example: partition array

Consider the following code to put the negative values at the beginning of array $b$:

```java
{{ 0 <= n <= b.length }}
i = k = 0;
{{ Inv: b[0], ..., b[k-1] < 0 <= b[k], ..., b[i-1] }}
while (i != n) {
    if (b[i] < 0) {
        swap b[i], b[k];
        k = k + 1;
    }
    i = i + 1;
}
{{ b[0], ..., b[k-1] < 0 <= b[k], ..., b[n-1] }}
```

- $I$ holds initially
- $I$ and $i = n$ implies postcondition
- $I$ is maintained by loop body