Loop Invariants: Part 2
Maintaining the Loop Invariant

• A claimed *loop invariant* is valid only if the loop body actually maintains the property, i.e., the loop invariant remains true at the end of each execution of the loop body

• To show this, you may assume:
  – The loop invariant is valid at the start of the loop body
  – The loop condition is true
The Loop Invariant Picture

To show the loop invariant true at this red point...

```c
while (test) {
    loop-body
}
```

...assume the invariant is true at this green point.
Isn’t This Reasoning Circular?

• To justify the assumption that the loop invariant holds just after the loop condition test, didn’t we argue that assumption was valid because the loop invariant holds just before the test?

• This is not circular reasoning but rather mathematical induction
  – See the confidence-building approach for reasoning about why recursion works
The Loop Invariant Picture

Show the loop invariant is true at this red point...

while (test) {
  loop-body
}

...which means it is true at this green point on the first iteration...
The Loop Invariant Picture

Show the loop invariant is true at this red point at the end of the first iteration...

\[
\text{while } (\text{test})\
\text{loop-body}
\]

...which means it is true at this green point on the second iteration...
Show the loop invariant is true at this red point at the end of the $k$-th iteration...

\begin{verbatim}
while (test) {
    loop-body
}
\end{verbatim}

...which means it is true at this green point on the $(k+1)$-st iteration...
Show the loop invariant is true at this red point at the end of the last iteration...

```java
while (test) {
    loop-body
}
```

...which means it is true at this green point when the loop terminates.
Example #2

```c
double power(double x, int p)
```

- Returns `x` to the power `p`.
- Requires: 
  \[ p > 0 \]
- Ensures: 
  \[ power = x^p \]
double result = 1.0;
double factor = x;
int pLeft = p;
/**
 * @updates result, factor, pLeft
 * @maintains
 * pLeft >= 0 and
 * result * factor^(pLeft) = x^(p)
 * @decreases
 * pLeft
 */
while (pLeft > 0) {
    ...
}
return result;
What are the values of the other variables here?
What Loop Body Would Work?

• Observation: $p_{Left}$ is positive at the start of the loop body, and the loop body has to decrease it
• How could you decrease $p_{Left}$?
Idea 1: Decrement $pLeft$

/**
 * @updates result, factor, pLeft
 * @maintains
 * $pLeft \geq 0$ and
 * $result \times factor^{(pLeft)} = x^{(p)}$
 * @decreases
 * $pLeft$
 */

while (pLeft > 0) {
    ...
    pLeft--;
}

7 January 2019
OSU CSE
The Rest of the Loop Body

• This is true at the start of the loop body (for each clause: why?):
  \[ p_{Left} \geq 0 \quad \text{and} \quad \text{result} \times \text{factor}^{p_{Left}} = x^{p} \quad \text{and} \quad p_{Left} > 0 \]

• This has to be true at the end of the loop body (for each clause: why?):
  \[ p_{Left} - 1 \geq 0 \quad \text{and} \quad \text{result} \times \text{factor}^{p_{Left} - 1} = x^{p} \]
The Rest of the Loop Body

• This is true at the start of the loop body (why?):

\[ pLeft \geq 0 \quad \text{and} \quad \text{result} \times \text{factor}^{pLeft} = x^{p} \quad \text{and} \quad pLeft > 0 \]

• This has to be true at the end of the loop body (why?):

\[ pLeft - 1 \geq 0 \quad \text{and} \quad \text{result} \times \text{factor}^{pLeft - 1} = x^{p} \]

Since \( x \) and \( p \) do not change in the loop (why?), the two circled expressions must be equal at the end of the loop body.
The Rest of the Loop Body

- We need to update \texttt{result} from \texttt{result}_i to \texttt{result}_f, and/or update \texttt{factor} from \texttt{factor}_i to \texttt{factor}_f, to make this true:

\[
\texttt{result}_i \times \texttt{factor}_i^{(pLeft)} = \texttt{result}_f \times \texttt{factor}_f^{(pLeft - 1)}
\]

- How could you do that?
The Rest of the Loop Body

• We need to update `result` from $\text{result}_i$ to $\text{result}_f$, and/or update `factor` from $\text{factor}_i$ to $\text{factor}_f$, to make this true:

$$\text{result}_i \times \text{factor}_i^{(\text{pLeft})} = \text{result}_f \times \text{factor}_f^{(\text{pLeft} - 1)}$$

• How could you do that?

One line of code that updates `result`:

```
result *= factor;
```
Idea 2: Halve $p_{Left}$

```c
/**
 * @updates result, factor, pLeft
 * @maintains
 * $p_{Left} \geq 0$ and
 * $result \times factor^{(p_{Left})} = x^{(p)}$
 * @decreases
 * $p_{Left}$
 */
while (pLeft > 0) {
    ...
    pLeft /= 2;
}
```
The Rest of the Loop Body

• This is true at the start of the loop body (for each clause: why?):
  \[ pLeft \geq 0 \quad \text{and} \quad \text{result} \times \text{factor}^{pLeft} = x^p \quad \text{and} \quad pLeft > 0 \]

• This has to be true at the end of the loop body (for each clause: why?):
  \[ pLeft/2 \geq 0 \quad \text{and} \quad \text{result} \times \text{factor}^{pLeft/2} = x^p \]
The Rest of the Loop Body

• We need to update result from result\textsubscript{i} to result\textsubscript{f}, and/or update factor from factor\textsubscript{i} to factor\textsubscript{f}, to make this true:

\[
result\textsubscript{i} \times factor\textsubscript{i}^{(pLeft)} = result\textsubscript{f} \times factor\textsubscript{f}^{(pLeft/2)}
\]

• How can you do that?
  – Remember: pLeft may be even or odd, but start with the simpler case where it is even