Recitation 5

Tyler Chan

February 4, 2025

Agenda

- Quiz 2 Review
- Loop Invariants
- Decrementing Functions
- Dafny Installation
- Quiz 3

Quiz 2 Review: Q2

Question 2. (25 pts., 10 pts. each part) Consider the following set of pre and postconditions for a method:

```
boolean is_float(String str_float)
//   Precondition: str_float != null and every character in str_float is one
//   of the following characters: '0'...'9', '.', '+', '-', 'e', or 'E'
//
//   Postcondition: The value returned by the
//   function is true if str_float represents a valid floating-point number;
//   otherwise the value returned by the function is false.
{
   ...  // Implementation of is_float()
}
```

Select one:

- true
- false
- null
- The result is undetermined
- An exception is thrown
- Code that calls is_float() would not compile.
- None of the above

- is_float(2.e-5)
- is_float(2.e-5.5)
- is_float(2f)
- is_float(null)

Quiz 2 Review: Q4

Question 4. (25 pts.) Find the strongest postcondition for the following set of statements. Show all intermediate conditions. Simplify your answer as much as possible but avoid making your conditions weaker. Only keep relevant variables. Assume all variables are integers. $\mathbb{N}$ denotes the set of all natural numbers.

```
{ (x%10 = 0) ∧ (x ≥ 5) ∧ (y = x) ∧ (z > 3) ∧ (w ∈ ℕ) ∧ (u > 0) } // This is the precondition.
z = z * x;
{ ---------------------------------------------------------------------- }
x = x + 1;
{ ---------------------------------------------------------------------- }
w = −w;
{ ---------------------------------------------------------------------- }
v = v % 10;
// This should be the strongest postcondition.
{ ---------------------------------------------------------------------- }
```

```
{ (x%10 = 0) ∧ (x ≥ 5)  ∧ (y = x) ∧ (z > 3) ∧ (w ∈ N) ∧ (u > 0) }
{ (x%10 = 0) ∧ (x ≥ 10) ∧ (y = x) ∧ (z > 3) ∧ (w ∈ N) ∧ (u > 0)}
z = z * x ;
{(x%10 = 0) ∧ (x ≥ 10) ∧ (y = x) ∧ (z >= 40) ∧ (z % 10 = 0) (w ∈ N) ∧ (u > 0)}
x = x + 1 ;
{(x%10 = 1) ∧ (x ≥ 11) ∧ (y = x-1) ∧ (z >= 40) ∧ (z % 10 = 0) ∧ (w ∈ N) ∧ (u > 0)}
w = -w;
{(x%10 = 1) ∧ (x ≥ 11) ∧ (y = x-1) ∧ (z >= 40) ∧ (z % 10 = 0) ∧ (-w ∈ N) ∧ (u > 0)}
v = v % 10;
{(x%10 = 1) ∧ (x ≥ 11) ∧ (y = x-1) ∧ (z >= 40) ∧ (z % 10 = 0) ∧ (-w ∈ N) ∧ (u > 0)}
or
{(x%10 = 1) ∧ (x ≥ 11) ∧ (y = x-1) ∧ (z >= 40) ∧ (z % 10 = 0) ∧ (-w ∈ N) ∧ (u > 0) ∧ (0 <= v <= 9}
```

Loop Invariants

- Loop Invariants are conditions that must hold at every iteration of a loop.
- Allows us to do forward reasoning with loops without having to reason about every step.

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

1. Come up with a loop invariant.

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

1. Come up with a loop invariant.
2. Prove base case holds (before the loop).

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

1. Come up with a loop invariant.
2. Prove base case holds (before the loop).
3. Prove loop invariant holds at each iteration.

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

1. Come up with a loop invariant.
2. Prove base case holds (before the loop).
3. Prove loop invariant holds at each iteration.
   - Assume the invariant holds for $(k-1)$th iteration.

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

1. Come up with a loop invariant.
2. Prove base case holds (before the loop).
3. Prove loop invariant holds at each iteration.
   - Assume the invariant holds for $(k-1)$th iteration.
   - Show that the invariant holds for kth iteration based on changes made to variables in the loop.

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

1. Come up with a loop invariant.
2. Prove base case holds (before the loop).
3. Prove loop invariant holds at each iteration.
   - Assume the invariant holds for $(k-1)$th iteration.
   - Show that the invariant holds for kth iteration based on changes made to variables in the loop.
4. Show that the invariant proves the post-condition at termination.

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Pre-condition:

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Pre-condition: {`n >= 0`}

Loop Invariant Example 1

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(\mathit{result} = (i-1)!) \land (i \leq n+1)$
- Base case:
  - $\mathit{result} = (i-1)!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$
  - Prove the other condition:
  - $i \leq n+1$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$
  - Prove the other condition:
  - $i \leq n+1$
  - $1 \leq n+1$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$
  - Prove the other condition:
  - $i \leq n+1$
  - $1 \leq n+1$
  - $0 \leq n$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$
  - Prove the other condition:
  - $i \leq n+1$
  - $1 \leq n+1$
  - $0 \leq n$
  - How to show this is true? re-condition!

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$
  - Prove the other condition:
  - $i \leq n+1$
  - $1 \leq n+1$
  - $0 \leq n$
  - How to show this is true? Pre-condition!

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$
- Base case:
  - $result = (i-1)!$
  - $1 = (1-1)!$
  - $1 = 0! \rightarrow True$
  - Prove the other condition:
  - $i \leq n+1$
  - $1 \leq n+1$
  - $0 \leq n$
  - How to show this is true? Pre-condition!
  - $0 \leq n = n \geq 0$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

- Induction:

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(\mathit{result} = (i-1)!) \land (i \leq n+1)$

- Induction:
  - Assume $\{\mathit{result}_{k-1} = ((k-1)-1)!\}$ is True.

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(\mathit{result} = (i-1)!) \wedge (i \leq n+1)$

- Induction:
  - Assume $\{\mathit{result}_{k-1} = ((k-1)-1)!\}$ is True.
  - $\mathit{result}_k = \mathit{result}_{k-1} * (k-1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(\mathit{result} = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{\mathit{result}_{k-1} = ((k-1)-1)!\}$ is True.
  - $\mathit{result}_k = \mathit{result}_{k-1} * (k-1)$
  - $\mathit{result}_k = ((k-1)-1)! * (k-1)$
  - $\mathit{result}_k = (k-2)! * (k-1)$
  - $\mathit{result}_k = (k-1)!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$
  - $result_k = (k-1)!$
  - Next condition says: $\{i_{k-1} \leq n+1\}$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$
  - $result_k = (k-1)!$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$
  - $result_k = (k-1)!$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.
  - $i_{k-1} < n+1$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$
  - $result_k = (k-1)!$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.
  - $i_{k-1} < n+1$
  - $i_{k-1} + 1 < n+2$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \land (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$
  - $result_k = (k-1)!$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.
  - $i_{k-1} < n+1$
  - $i_{k-1} + 1 < n+2$
  - $i_{k-1} + 1 \leq n+1$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}
- Loop invariant: $(result = (i-1)!) \wedge (i \leq n+1)$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Induction:
  - Assume $\{result_{k-1} = ((k-1)-1)!\}$ is True.
  - $result_k = result_{k-1} * (k-1)$
  - $result_k = ((k-1)-1)! * (k-1)$
  - $result_k = (k-2)! * (k-1)$
  - $result_k = (k-1)!$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.
  - $i_{k-1} < n+1$
  - $i_{k-1} + 1 < n+2$
  - $i_{k-1} + 1 \leq n+1$
  - $i_k \leq n+1$, which is the loop invariant

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:
    - Since the loop terminates, $i \geq n + 1$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:
  - Since the loop terminates, $i \geq n + 1$
  - $(i \geq n + 1) \wedge (i \leq n + 1) \wedge result = (i - 1)!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:
  - Since the loop terminates, $i \geq n + 1$
  - $(i \geq n + 1) \wedge (i \leq n + 1) \wedge \mathit{result} = (i - 1)!$
  - $(i = n + 1) \wedge \mathit{result} = (i - 1)!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:
  - Since the loop terminates, $i \geq n+1$
  - $(i \geq n+1) \wedge (i \leq n+1) \wedge \mathit{result} = (i-1)!$
  - $(i = n+1) \wedge \mathit{result} = (i-1)!$
  - $\mathit{result} = (n+1-1)!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:
  - Since the loop terminates, $i \geq n + 1$
  - $(i \geq n + 1) \wedge (i \leq n + 1) \wedge \mathit{result} = (i - 1)!$
  - $(i = n + 1) \wedge \mathit{result} = (i - 1)!$
  - $\mathit{result} = (n + 1 - 1)!$
  - $\mathit{result} = n!$

Loop Invariant Example 1

- Pre-condition: {`n >= 0`}
- Post-condition: {`result = n!`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Termination:
  - Since the loop terminates, $i \geq n + 1$
  - $(i \geq n + 1) \wedge (i \leq n + 1) \wedge \textit{result} = (i - 1)!$
  - $(i = n + 1) \wedge \textit{result} = (i - 1)!$
  - $\textit{result} = (n + 1 - 1)!$
  - $\textit{result} = n!$
  - Post-condition holds!

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- What is the loop invariant?
- Prove the loop invariant is true at the start of the first iteration.

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- What is the loop invariant?
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$
- Prove the loop invariant is true at the start of the first iteration.

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: $\text{sum} = \frac{n(n+1)}{2}$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- What is the loop invariant?
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$
- Prove the loop invariant is true at the start of the first iteration.
  - $\text{sum} = \sum_{j=1}^{0} j$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- What is the loop invariant?
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$
- Prove the loop invariant is true at the start of the first iteration.
  - sum $= \sum_{j=1}^{0} j$
  - $0 = 0$
  - True

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

- Induction:

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$
  - $\text{sum}_k = 1 + \cdots + (i_{k-1} - 1) + i_{k-1}$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$
  - $\text{sum}_k = 1 + \cdots + (i_{k-1} - 1) + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}} j$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: $\text{sum} = \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$
  - $\text{sum}_k = 1 + \cdots + (i_{k-1} - 1) + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}} j$
  - $\text{sum}_k = \sum_{j=1}^{i_k - 1} j$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$
  - $\text{sum}_k = 1 + \cdots + (i_{k-1} - 1) + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}} j$
  - $\text{sum}_k = \sum_{j=1}^{i_k - 1} j$
  - Next condition says: $\{i_{k-1} \leq n+1\}$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$
  - $\text{sum}_k = 1 + \cdots + (i_{k-1} - 1) + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}} j$
  - $\text{sum}_k = \sum_{j=1}^{i_k - 1} j$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Induction:
  - Assume $k-1$ is true: $\text{sum}_{k-1} = \sum_{j=1}^{i_{k-1}-1} j$
  - $\text{sum}_k = \text{sum}_{k-1} + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}-1} j + i_{k-1}$
  - $\text{sum}_k = 1 + \cdots + (i_{k-1} - 1) + i_{k-1}$
  - $\text{sum}_k = \sum_{j=1}^{i_{k-1}} j$
  - $\text{sum}_k = \sum_{j=1}^{i_k-1} j$
  - Next condition says: $\{i_{k-1} \leq n+1\}$
  - But: $\{i_{k-1} < n+1\}$ must be True because we are in the loop.
  - $i_k = i_{k-1} + 1 \leq n+1$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \geq n+1) \wedge (i \leq n+1)$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \geq n+1) \wedge (i \leq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i = n+1)$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \geq n+1) \land (i \leq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i = n+1)$
  - sum $= \sum_{j=1}^{(n+1)-1} j$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i \geq n+1) \land (i \leq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \land (i = n+1)$
  - $\text{sum} = \sum_{j=1}^{(n+1)-1} j$
  - $\text{sum} = \sum_{j=1}^{n} j$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: $\text{sum} = \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \geq n+1) \wedge (i \leq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i = n+1)$
  - $\text{sum} = \sum_{j=1}^{(n+1)-1} j$
  - $\text{sum} = \sum_{j=1}^{n} j$
  - $\text{sum} = \frac{n(n+1)}{2}$

Loop Invariant Example 2

- Pre-condition: $n \geq 0$
- Post-condition: sum $= \frac{n(n+1)}{2}$
- Loop Invariant: $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \leq n+1)$

```
int sum = 0;
int i = 0;
while (i <= n) {
    sum = sum + i;
    i = i + 1
}
```

- Termination:
  - Since the loop terminates, $(i \geq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i \geq n+1) \wedge (i \leq n+1)$
  - $(\text{sum} = \sum_{j=1}^{i-1} j) \wedge (i = n+1)$
  - sum $= \sum_{j=1}^{(n+1)-1} j$
  - sum $= \sum_{j=1}^{n} j$
  - sum $= \frac{n(n+1)}{2}$
  - Post-condition holds!

Decrementing Function

- We also need to prove that our loops terminate.

Decrementing Function

- We also need to prove that our loops terminate.
- This is called a **decrementing function**.

Decrementing Function

- We also need to prove that our loops terminate.
- This is called a **decrementing function**.
- Decrementing functions are functions whose value is always decreasing.

Decrementing Function

- We also need to prove that our loops terminate.
- This is called a **decrementing function**.
- Decrementing functions are functions whose value is always decreasing.
- The loop must terminate at it's minimum value.

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Decrementing function:

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Decrementing function:
  - $D = n - i + 1$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Decrementing function:
  - $D = n - i + 1$
- Initally:

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- Decrementing function:
  - $D = n - i + 1$
- Initally:
  - $D = n - 1 + 1$
  - $D = n \geq 0$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

- During the loop:

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.
- When $D = 0$, the loop terminates.

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.
- When $D = 0$, the loop terminates.
  - We know that $i = n + 1$ at the end of the loop (from LI proof).

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.
- When $D = 0$, the loop terminates.
  - We know that $i = n + 1$ at the end of the loop (from LI proof).
  - $D = n - i + 1$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.
- When $D = 0$, the loop terminates.
  - We know that $i = n + 1$ at the end of the loop (from LI proof).
  - $D = n - i + 1$
  - $D = n - (n + 1) + 1$

Decrementing Function Example 1

- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.
- When $D = 0$, the loop terminates.
  - We know that $i = n + 1$ at the end of the loop (from LI proof).
  - $D = n - i + 1$
  - $D = n - (n + 1) + 1$
  - $D = n - n - 1 + 1 = 0$

Decrementing Function Example 1

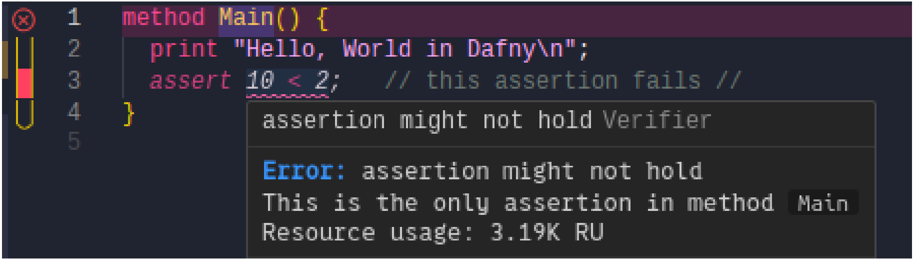
- Pre-condition: {`n >= 0`}

```
def factorial(n):
    result = 1
    i = 1
    while i <= n:
        result *= i
        i += 1
    return result
```

- During the loop:
  - $D_{k-1} = n - i_{k-1} + 1$
  - $i_k = i_{k-1} + 1$
  - $D_k = n - i_k + 1$
  - $D_k = n - (i_{k-1} + 1) + 1$
  - $D_k = n - i_{k-1} - 1 + 1$
  - $D_k = D_{k-1} - 1$
  - D does decrease after 1 iteration.
- When $D = 0$, the loop terminates.
  - We know that $i = n + 1$ at the end of the loop (from LI proof).
  - $D = n - i + 1$
  - $D = n - (n + 1) + 1$
  - $D = n - n - 1 + 1 = 0$
  - The loop terminates, and the decrementing function hits its minimum value.

Dafny Installation

- https://dafny.org/latest/Installation
- I would just install the VSCode extension, since it's easy to install.
- Write a HelloWorld.dfy file and hit run (F5).



After Recitation Stuff

If you are using VSCODE:

- If it prompts you that you are missing DOTNET, install the 5.0 (or higher) DOTNET SDK
- Restart VSCode
- On the bottom right, there should be a textbox and you should install Dafny **4.9.1**
- Restart VSCode

Quiz 3

Do quiz 3 now on Submitty.