1. Computation via Verification Condition Generator

```
vc[[ y = 0;
     {x >= 0 & x + y = x0}
     while (x > 0) {
        x = x - 1;
        y = y + 1
} ]](x >= 0 & x = x0, y = x0) = {Composition}

vc[[ y = 0 ]](
     vc[[ {x >= 0 & x + y = x0}
          while (x > 0) {x = x - 1; y = y + 1}
     ]](x >= 0 & x = x0, y = x0)
) = {Annotated-While}

vc[[ y = 0 ]](
     let (ψ', ϕ') = vc[[
          x = x - 1; y = y + 1
     ]](x >= 0 & x + y = x0, true)
     in
     (x >= 0 & x + y = x0, y = x0 & ϕ'
     & (x >= 0 & x + y = x0 & x > 0 ==> ψ')
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0))
) = {Composition}
```

```
vc[[ y = 0 ]](
     let (ψ', ϕ') = vc[[  x = x - 1 ]](
          vc[[ y = y + 1 ]](x >= 0 & x + y = x0, true)
     )
     in
     (x >= 0 & x + y = x0, y = x0 & ϕ'
     & (x >= 0 & x + y = x0 & x > 0 ==> ψ')
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0))
) = {Assignment}

vc[[ y = 0 ]](
     let (ψ', ϕ') = vc[[  x = x - 1 ]](
          x >= 0 & x + (y + 1) = x0, true
     )
     in
     (x >= 0 & x + y = x0, y = x0 & ϕ'
     & (x >= 0 & x + y = x0 & x > 0 ==> ψ')
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0))
) = {Assignment}

vc[[ y = 0 ]](
     let (ψ', ϕ') = ((x - 1) >= 0 & (x - 1) + (y + 1) = x0, true)
     in
     (x >= 0 & x + y = x0, y = x0 & ϕ'
     & (x >= 0 & x + y = x0 & x > 0 ==> ψ')
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0))
) = {Let-In-Substitution}
```

```
vc[[ y = 0 ]](
     x >= 0 & x + y = x0, y = x0 & true
     & (x >= 0 & x + y = x0 & x > 0
          ==> (x - 1) >= 0 & (x - 1) + (y + 1) = x0)
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0)
) = {Assignment}

(
     x >= 0 & x + 0 = x0, y = x0 & true
     & (x >= 0 & x + y = x0 & x > 0
          ==> (x - 1) >= 0 & (x - 1) + (y + 1) = x0)
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0)
) = {Removing Redundancies}

(
     x >= 0 & x = x0,
     y = x0
     & (x >= 0 & x + y = x0 & x > 0
          ==> (x - 1) >= 0 & x + y = x0)
     & (x >= 0 & x + y = x0 & !(x > 0) ==> x >= 0 & x = x0)
) = {Simplification of Implication Assumptions}

(
     x >= 0 & x = x0,
     y = x0
     & (x + y = x0 & x > 0 ==> (x - 1) >= 0 & x + y = x0)
     & (x = 0 & x + y = x0 ==> x >= 0 & x = x0)
) = {Implications Evaluate to True}
```

```
(
     x >= 0 & x = x0,
     y = x0
) = vc[[ Program ]](x >= 0 & x = x0, y = x0)
```

2. Computation via Weakest Precondition

Translated program in the Intermediate Language:

```
y = 0;
assert x >= 0 & x + y = x0;
havoc [x, y];
assume x >= 0 & x + y = x0;
(
     assume x > 0;
     x = x - 1;
     y = y + 1;
     assert x >= 0 & x + y = x0;
     assume false
     []
     assume x <= 0
)
```

Computation of verification conditions, adding pre- and post-conditions:

```
wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
```

```
     (
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0;
          assume false
          []
          assume x <= 0
     );
     assert y = x0
     ,
     true
) = {Composition}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     (
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0;
          assume false
          []
```

```
          assume x <= 0
     )
     ,
     wp(assert y = x0, true)
) = {Assert}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     (
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0;
          assume false
          []
          assume x <= 0
     )
     ,
     y = x0 & true
) = {Composition}

wp(
     assume x >= 0 & x = x0;
```

```
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          (
               assume x > 0;
               x = x - 1;
               y = y + 1;
               assert x >= 0 & x + y = x0;
               assume false
               []
               assume x <= 0
          )
          ,
          y = x0 & true
     )
) = {Choice}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
```

```
     wp(
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0;
          assume false
          ,
          y = x0 & true
     )
     &
     wp(assume x <= 0, y = x0 & true)
) = {Assume}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0;
          assume false
          ,
```

```
          y = x0 & true
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Composition}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0;
          wp(assume false, y = x0 & true)
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Assume}

wp(
     assume x >= 0 & x = x0;
     y = 0;
```

```
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1;
          y = y + 1;
          assert x >= 0 & x + y = x0,
          false ==> y = x0 & true
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Composition}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1;
          y = y + 1,
          wp(assert x >= 0 & x + y = x0, false ==> y = x0 & true)
```

```
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Assert}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1;
          y = y + 1,
          x >= 0 & x + y = x0 & false ==> y = x0 & true
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Composition}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
```

```
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1,
          wp(y = y + 1,
             x >= 0 & x + y = x0 & false ==> y = x0 & true)
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Assignment}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0;
          x = x - 1,
          x >= 0 & x + (y + 1) = x0 & false ==> (y + 1) = x0 & true
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Composition}
```

```
wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0,
          wp(x = x - 1,
          x >= 0 & x + (y + 1) = x0 & false ==> (y + 1) = x0 & true)
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Assignment}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     wp(
          assume x > 0,
          (x - 1) >= 0 & (x - 1) + (y + 1) = x0
               & false ==> (y + 1) = x0 & true)
```

```
     &
     (x <= 0 ==> y = x0 & true)
) = {Assume}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y];
     assume x >= 0 & x + y = x0;
     ,
     (
          x > 0 ==>
               (x - 1) >= 0 & (x - 1) + (y + 1) = x0
               & false ==> (y + 1) = x0 & true
     )
     &
     (x <= 0 ==> y = x0 & true)
) = {Composition}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y],
     wp(
          assume x >= 0 & x + y = x0
```

```
          ,
          (
               x > 0 ==>
                    (x - 1) >= 0 & (x - 1) + (y + 1) = x0
                    & false ==> (y + 1) = x0 & true
          )
          &
          (x <= 0 ==> y = x0 & true)
     )
) = {Assume}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0;
     havoc [x, y],
     x >= 0 & x + y = x0 ==>
          (
               x > 0 ==>
                    (x - 1) >= 0 & (x - 1) + (y + 1) = x0
                    & false ==> (y + 1) = x0 & true
          )
     &
     (x <= 0 ==> y = x0 & true)
) = {Composition}
```

```
wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0,
     wp(havoc [x, y],
          x >= 0 & x + y = x0 ==>
               (
                    x > 0 ==>
                         (x - 1) >= 0 & (x - 1) + (y + 1) = x0
                         & false ==> (y + 1) = x0 & true
               )
          &
          (x <= 0 ==> y = x0 & true)
     )
) = {Havoc with x’ and y’ being new variables}

wp(
     assume x >= 0 & x = x0;
     y = 0;
     assert x >= 0 & x + y = x0,
     x’ >= 0 & x’ + y’ = x0 ==>
          (
               x’ > 0 ==>
                    (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
                    & false ==> (y’ + 1) = x0 & true
          )
     &
```

```
     (x’ <= 0 ==> y’ = x0 & true)
) = {Composition}

wp(
     assume x >= 0 & x = x0;
     y = 0,
     wp(assert x >= 0 & x + y = x0,
          x’ >= 0 & x’ + y’ = x0 ==>
               (
                    x’ > 0 ==>
                         (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
                         & false ==> (y’ + 1) = x0 & true
               )
          &
          (x’ <= 0 ==> y’ = x0 & true)
     )
) = {Assert}

wp(
     assume x >= 0 & x = x0;
     y = 0,
     x >= 0 & x + y = x0
     &
     x’ >= 0 & x’ + y’ = x0 ==>
          (
               x’ > 0 ==>
                    (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
```

```
                    & false ==> (y’ + 1) = x0 & true
          )
     &
     (x’ <= 0 ==> y’ = x0 & true)
) = {Composition}

wp(
     assume x >= 0 & x = x0,
     wp(y = 0,
          x >= 0 & x + y = x0
          &
          x’ >= 0 & x’ + y’ = x0 ==>
               (
                    x’ > 0 ==>
                         (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
                         & false ==> (y’ + 1) = x0 & true
               )
          &
          (x’ <= 0 ==> y’ = x0 & true)
     )
) = {Assignment}

wp(
     assume x >= 0 & x = x0,
     x >= 0 & x + 0 = x0
     &
     x’ >= 0 & x’ + y’ = x0 ==>
```

```
          (
               x’ > 0 ==>
                    (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
                    & false ==> (y’ + 1) = x0 & true
          )
     &
     (x’ <= 0 ==> y’ = x0 & true)
) = {Assume}

x >= 0 & x = x0 ==> (
     x >= 0 & x + 0 = x0
     &
     x’ >= 0 & x’ + y’ = x0 ==>
          (
               x’ > 0 ==>
                    (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
                    & false ==> (y’ + 1) = x0 & true
          )
     &
     (x’ <= 0 ==> y’ = x0 & true)
) = {First conjunct removed since it is equivalent to the assumption}

x >= 0 & x = x0 ==> (
     x’ >= 0 & x’ + y’ = x0 ==>
          (
               x’ > 0 ==>
                    (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
```

```
                    & false ==> (y’ + 1) = x0 & true
          )
     &
     (x’ <= 0 ==> y’ = x0 & true)
) = {Implication from false is true, it is redundant in conjunction}

x >= 0 & x = x0 ==> (
     x’ >= 0 & x’ + y’ = x0 ==> (
               x’ > 0 ==> (x’ - 1) >= 0 & (x’ - 1) + (y’ + 1) = x0
          )
     &
     (x’ <= 0 ==> y’ = x0 & true)
) = {Inner most implication evaluates to true, redundant}

x >= 0 & x = x0 ==> (
     x’ >= 0 & x’ + y’ = x0 ==> ((x’ - 1) + (y’ + 1) = x0)
     &
     (x’ <= 0 ==> y’ = x0 & true)
) = {First conjunction in innermost conclusion evaluates to true}

x >= 0 & x = x0 ==> (
     x’ >= 0 & x’ + y’ = x0 ==> (x’ <= 0 ==> y’ = x0 & true)
) = {We get x’ = 0, hence the implication evaluates to true}

(x >= 0 & x = x0 ==> true) {= true}
```

Hence, (|x >= 0 & x = x0|) *Program* (|y = x0|) is valid.

3. Comparison of Results

Resulting formula from Verification Condition Generator:

```
(
     x >= 0 & x = x0,
     y = x0
) = vc[[ Program ]](x >= 0 & x = x0, y = x0)
```

Resulting formula from Weakest Precondition Calculation:

```
(x >= 0 & x = x0 ==> true) {= true}
```

I am not sure exactly what aspects to compare, I think these two methods achieve their goals through different means. In the first method we get the pre- and post- conditions that exactly match our contract. So, we know our program is correct. In the second method we assume our contract and get the weakest precondition to be true, which means our program is always valid under the assumed contract.

Thus, we get different results but both tell us that our program is correct under the given contract.