```
/**

 */
 function fib (n:nat): nat
 {
     if n==0 then 0 else
     if n==1 then 1 else fib(n-2) + fib(n-1)
 }

 method ComputeFib'' (n:nat) returns (b:nat)
    ensures (b == fib(n))
    {
        //introduce local variable
        var i, c;
        b,c,i := CF2(n);
    }

predicate Inv(n:nat, b:nat, c:nat, i:nat)
{
    0 <= i <= n && b == fib(i) && c == fib(i+1)
}

/**
In Dafny, (b:nat, c:nat, i:nat) this is the frame - the variables we can change their value
 */
 method CF2 (n:nat) returns (b:nat, c:nat, i:nat)
    ensures b == fib(n) //post
    {
        //strengthen post condition
        b,c,i := CF3(n);
        L3(n,b,c,i);
    }

 lemma L3 (n:nat, b:nat, c:nat, i:nat)
    requires Inv(n,b,c,i) && i>=n
    ensures b == fib(n)

 method CF3 (n:nat) returns (b:nat, c:nat, i:nat)
    ensures Inv(n,b,c,i) && i>=n //post', we want post' ==> post, using Lemma3. i<= n : the not of the guard
    {
        b,c,i := CF4a(n);
        b,c,i := CF4b(n,b,c,i);
    }

//initialization = establish the invariant
 method CF4a (n:nat) returns (b:nat, c:nat, i:nat)
    ensures Inv(n,b,c,i) // mid, cant have 0-named variables
    {
        //assignment
        L4a(n,b,c,i);
        i,b,c := 0,0,1;
    }
lemma L4a(n:nat,b:nat, c:nat, i:nat)
    ensures Inv(n,0,1,0) //subsitution
    {}


method CF4b(n:nat,b0:nat, c0:nat, i0:nat) returns (b:nat, c:nat, i:nat)
    requires Inv(n,b0,c0,i0) //mid (in terms of the initial variables)
    ensures Inv(n,b,c,i) && i≥n //post condition
    {
        b,c,i := b0,c0,i0; //convention, else = garbage value
        while i < n
            invariant Inv(n,b,c,i)
            decreases n-i
            {
                b,c,i := CF5(n,b,c,i);
            }
    }
method CF5(n:nat,b0:nat, c0:nat, i0:nat) returns (b:nat, c:nat, i:nat)
    requires Inv(n,b0,c0,i0) && i0 < n
    ensures Inv(n,b,c,i) && 0 ≤ n - i < n-i0
    {
        b,c,i := b0,c0,i0;
        L5(n,b,c,i);
        b,c,i := c,b+c,i+1;
    }
lemma L5(n:nat,b:nat, c:nat, i:nat)
    requires Inv(n,b,c,i) && i < n
    ensures Inv(n,c,b+c,i+1) && 0 ≤ n - (i+1) < n-i
    {}

```