

```

1 /**
2
3 */
4 function fib (n:nat): nat
5 {
6     if n==0 then 0 else
7     if n==1 then 1 else fib(n-2) + fib(n-1)
8 }
9
10 method ComputeFib'' (n:nat) returns (b:nat)
11     ensures (b = fib(n))
12     {
13         //introduce local variable
14         var i, c;
15         b,c,i := CF2(n);
16     }
17
18 predicate Inv(n:nat, b:nat, c:nat, i:nat)
19 {
20     0 ≤ i ≤ n && b = fib(i) && c = fib(i+1)
21 }
22
23 /**
24 In Dafny, (b:nat, c:nat, i:nat) this is the frame - the variables we can change their
value
25 */
26 method CF2 (n:nat) returns (b:nat, c:nat, i:nat)
27     ensures b = fib(n) //post
28     {
29         //strengthen post condition
30         b,c,i := CF3(n);
31         L3(n,b,c,i);
32     }
33
34 lemma L3 (n:nat, b:nat, c:nat, i:nat)
35     requires Inv(n,b,c,i) && i ≥ n
36     ensures b = fib(n)
37
38 method CF3 (n:nat) returns (b:nat, c:nat, i:nat)
39     ensures Inv(n,b,c,i) && i ≥ n //post', we want post' ⇒ post, using Lemma3. i ≤ n :
the not of the guard
40     {
41         b,c,i := CF4a(n);
42         b,c,i := CF4b(n,b,c,i);
43     }
44
45 //initialization = establish the invariant
46 method CF4a (n:nat) returns (b:nat, c:nat, i:nat)
47     ensures Inv(n,b,c,i) // mid, cant have 0-named variables
48     {
49         //assignment

```

```

50     L4a(n,b,c,i);
51     i,b,c := 0,0,1;
52 }
53 lemma L4a(n:nat,b:nat, c:nat, i:nat)
54   ensures Inv(n,0,1,0) //subsitution
55   {}
56
57
58 method CF4b(n:nat,b0:nat, c0:nat, i0:nat) returns (b:nat, c:nat, i:nat)
59   requires Inv(n,b0,c0,i0) //mid (in terms of the initial variables)
60   ensures Inv(n,b,c,i) && i ≥ n //post condition
61   {
62     b,c,i := b0,c0,i0; //convention, else = garbage value
63     while i < n
64       invariant Inv(n,b,c,i)
65       decreases n-i
66       {
67         b,c,i := CF5(n,b,c,i);
68       }
69   }
70 method CF5(n:nat,b0:nat, c0:nat, i0:nat) returns (b:nat, c:nat, i:nat)
71   requires Inv(n,b0,c0,i0) && i0 < n
72   ensures Inv(n,b,c,i) && 0 ≤ n - i < n-i0
73   {
74     b,c,i := b0,c0,i0;
75     L5(n,b,c,i);
76     b,c,i := c,b+c,i+1;
77   }
78 lemma L5(n:nat,b:nat, c:nat, i:nat)
79   requires Inv(n,b,c,i) && i < n
80   ensures Inv(n,c,b+c,i+1) && 0 ≤ n - (i+1) < n-i
81   {}
82

```