Blue Proof

We need to prove that after the execution of `dbl`, every element at index j in the list satisfies lst[j] = 2 * lst″[j], where lst″ represents the original input list.

Loop Invariant:

We define the loop invariant P(n) as follows: immediately before the n-th iteration of the loop , let the current index be `i` . Then, for every index j satisfying $0 \leq j < i$, the value in the list has already been updated such that lst[j] = 2 * lst″[j]; moreover, for every index j such that $j \geq i$, the value remains unchanged, meaning lst[j] = lst″[j].

Initialization:

Before the loop starts, i is initialized to 0. Since there are no indices j with $0 \leq j < 0$, the first part of the invariant holds vacuously.

Furthermore, the entire list (from index 0 onward) is unchanged, so lst[0:] equals the original list lst″. Thus, the invariant P(0) holds.

Maintenance:

Assume that just before a given iteration the invariant holds with the current index i (< len(lst)).

At this moment, for every j with $0 \leq j < i$, the list has already been updated correctly, and for all $j \geq i$, we have lst[j] = lst″[j]. In the iteration, the function updates lst[i] in line 4 by executing lst[i] = 2 * lst[i].

Because of the invariant, prior to this update, lst[i] equals lst″[i]. Therefore, after the update, lst[i] becomes 2 * lst″[i].

Then, i is incremented in line 5, so the invariant now holds for indices $0 \leq j < i + 1$ (all elements up to and including the updated element) while the remainder of the list is still unchanged. Thus, the invariant is maintained.

Termination:

The loop terminates when the condition in line 3 fails, meaning that i equals len(lst). At termination, the invariant guarantees that for every index j from 0 to len(lst) – 1, lst[j] = 2 * lst″[j].

This confirms that every element in the list has been correctly doubled. The function then ends, having established the desired property.