15-122: Principles of Imperative Computation

Recitation 3: A Strange Sort of Proof James Wu, Andrew Benson, Aaron Gutierrez

In class, we covered one quadratic sort, selection sort. Today, we'll look at the full correctness proof, from beginning to end, for another type of sort, known as insertion sort. This is somewhat more complex of a proof, so be sure to follow along carefully!

Insertion Sort

```
1  void sort(int[] A, int n)
2  //@requires 0 <= n && n <= \length(A);
3  //@ensures is_sorted(A, 0, n);
4  {
5     for (int i = 0; i < n; i++)
6     //@loop_invariant 0 <= i && i <= n;
7     //@loop_invariant is_sorted(A, 0, i);
8     {
9        int j = i;
10
11       while (j > 0 && A[j-1] > A[j])
12       //@loop_invariant 0 <= j && j <= i;
13       //@loop_invariant is_sorted(A, 0, j);
14       //@loop_invariant is_sorted(A, j, i+1);
15       //@loop_invariant le_segs(A, 0, j, A, j+1, i+1);
16       {
17          swap(A, j-1, j);
18          j--;
19       }
20    }
21 }
```

To proceed, we need to follow the same four steps we have used all semester to show correctness for a function with a loop. But when we get to the preservation step, the loop body is itself a loop, so we must repeat the processes.

1. Prove that, given the preconditions, the outer loop invariants initially hold.

In order to prove that the outer loop invariants are correct, we will need the negation of the inner loop guard along with the preservation and correctness of the inner loop invariants.

(a) Assume that the outer loop invariants hold for some iteration of the loop. Prove that the **inner** loop invariants initially hold on this iteration. You can imagine you are trying to show the loop invariants are true initially for the following block of code:

```
6 //@assert 0 <= i && i <= n;
7 //@assert is_sorted(A, 0, i);
8
9 int j=i;
10
11 while (j > 0 && A[j-1] > A[j])
12 //@loop_invariant 0 <= j && j <= i;
13 //@loop_invariant is_sorted(A, 0, j);
14 //@loop_invariant is_sorted(A, j, i+1);
15 //@loop_invariant le_segs(A, 0, j, A, j+1, i+1);
16 {
17    swap(A, j-1, j);
18    j--;
19 }
```

Line 12:
Line 13:
Line 14:
Line 15:

(b) Given that the inner loop invariants hold initially, prove that the **inner** loop invariants are preserved.

Line 12:

Line 13:

Line 14:

Line 15:

(c) Prove the inner loop terminates.

We successfully proved the inner loop invariants correct! Now all that's left is to prove the outer ones.

2. Prove the outer loop invariants are preserved, given that the inner loop invariants hold.

3. Prove that the outer loop terminates.

4. Finally, prove that the termination of the outer loop and the negation of the outer loop guard prove the postcondition. Whew! We're done!