S. KIMMEL

Loop Invariants: Prove loops are correct

```
setup
while (condition) {
    stuff
}
Great output
```

↙ Induction tailored to loops

Parts of Loop Invariant Proof

1. State Invariant: thing(s) that is true before & after each loop iteration
2. Base Case: Show invariant is true before loop starts.
3. Maintenance: Show if invariant is true before an iteration, it is true after an iteration
4. Termination: argue loop ends. Given status of invariant after final loop, argue great output.

Input : Array A of integers of length n

Output: Smallest value of A

```
1 min = A[1];
2 i = 2;
3 while i ≤ n do
4 |  if A[i] < min then
5 |  |  min = A[i]
6 |  end
7 |  i++;
8 end
9 return min;
```

Algorithm 1: `Smallest`(A)

S.KIMMEL

Loop Invariants

ex:

```
MIN(array A of length n)
• min = A[1]
• i = 2
• while (i ≤ n)
      if (A[i] < min ):
          min = A[i]
      i++
  return min
```

Loop Invariant

- min is minimum of A[1:i-1]

Base Case: Before loop starts: i=2, min = A[1].

- min is minimum of A[1:1]

S.KIMMEL

Maintenance

min = minimum $\{A[1:i-1]\}$ at start of loop. But
minimum $\{A[1:i]\}$ = minimum $\{A[1:i-1], A[i]\}$ = minimum $\{min, A[i]\}$,
which is what min becomes at the end of the loop.

Termination:
- loop terminates at $i = n+1$
- i increases, so termination will occur

Invariant: min = minimum $\{A[1:n]\}$

* Make sure your invariant gives you the result you want at end of loop *

Input : Array A of integers of length n

Output: Array containing sorted elements of A

```
1 for k = 1 to n − 1 do
2 |  for j = n to k + 1 do
3 |  |  if A[j] < A[j − 1] then
4 |  |  |  Swap A[j] and A[j − 1];
5 |  |  end
6 |  end
7 end
8 return A;
```

Algorithm 2: `BubbleSort`(A)