COMPUTER SCIENCE & ENGINEERING

EECS498-008
Formal Verification of Systems Software

Material and slides created by

Jon Howell and Manos Kapritsos

Detour to Imperativeland

```
lemma loop(target:nat) returns (result:nat)
    ensures result == target
{
  result := 0;
  while (result < target)
    invariant result <= target
  {
    result := result + 1;
  }
  return result;
}
```

Dafny needs an invariant to reason about the loop's body

Detour to Imperativeland

```
predicate IsMaxIndex(a:seq<int>, x:int) {
  && 0 <= x < |a|
  && (forall i | 0 <= i < |a| :: a[i] <= a[x])
}
```

Note that the order of conjuncts matters!

And the same is true for ensures/requires: their order matters

Imperativeland

```
method findMaxIndex(a:seq<int>) returns (x:int)
  requires |a| > 0
  ensures IsMaxIndex(a, x)
{
  var i := 1;
  var ret := 0;
  while(i < |a|)
    invariant 0 <= i <= |a|
    invariant IsMaxIndex(a[..i], ret)
  {
    if(a[i] > a[ret]) {
      ret := i;
    }
    i := i + 1;
  }
  return ret;
}
```

```
predicate IsMaxIndex(a:seq<int>, x:int) {
  && 0 <= x < |a|
  && (forall i | 0 <= i < |a| :: a[i] <=
a[x])
}
```

Logistics

- You should all have access to autograder.io
  - Let me know ASAP if that is not the case
- Some students have conflicts with the lab on Friday
  - I will give you 48-hour access to the recording if you have a conflict
- Chapter 1 is released
  - Chapter 2 will follow soon

Chapter 1 progress

- **Some** of you have already submitted
- Pleeeeenty of time left, don’t worry… 😈

Autograder submissions: the RULES

- You may not use /* */ comments
- You must leave the existing /* */ comments in place
- You may only change text between /*{*/ and /*}*/
- You are not allowed to add axioms (or to otherwise trivialize the proof)

- You are given three submissions per day
- You are given three late day tokens throughout the semester

Example: exercise01.dfy

```
//#title Lemmas and assertions

lemma IntegerOrdering()
{
  // An assertion is a **static** check of a boolean expression -- a mathematical
truth.
  // This boolean expression is about (mathematical) literal integers.
  // Run dafny on this file. See where it fails. Fix it.
  assert /*{*/ 5 < 3 /*}*/;
}
```

Recursion: exporting ensures

```
function Evens(count:int) : (outseq:seq<int>)
  ensures forall idx :: 0<=idx<|outseq| ==> outseq[idx] == 2 * idx
{
  if count==0 then [] else Evens(count) + [2 * (count-1)]
}
```