Imperative programming in Dafny

Exercise: Insertion Sort  
Will Sonnex and Sophia Drossopoulou

The following exercise is about loops, loop invariants, ghost variables, and the discovery and use of lemmas in the proof correctness of imperative programs.

The Dafny code can be found in the file Arrays-Insertion\_Sort.dfy. The exercise sheet makes use of Dafny sequences described in <http://rise4fun.com/Dafny/tutorialcontent/Sequences>.

### **Part A**

The predicate sorted\_between expresses whether a sequence is sorted in ascending order between two indices, including the from index but not the to index.

predicate sorted\_between(a: array<int>, from:int, to:int)

requires a!= null; reads a;

{ forall i,j :: from<=i<j<to && 0<=i<j<a.Length ==> a[i]<=a[j] }

predicate sorted(a: array<int>)

requires a!= null; reads a;

{ sorted\_between(a,0,a.Length); }

### **Part B**

The function count returns the number of occurrences of x in the sequence s.

function count<T>(x: T, s: seq<T>): nat

{

if (|s| == 0)

then 0

else if x == s[0]

then 1 + count(x, s[1..])

else count(x, s[1..])

}

State and prove a lemma that says that the number of occurrences of a value in the concatenation of sequences s and t, is the sum of the occurrences in s and in t.

The predicate perm guarantees that sequence a Is a permutation of sequence b.

predicate perm<T>(a: seq<T>, b: seq<T>)

{

forall x :: count(x, a) == count(x, b)

}

State and prove a lemma that says that the relation perm is transitive, ie if a is a permutation of b, and b is a permutation of c, then a is a permutation of c.

### **Part C**

The specification of the following method InsertionSort, prescribes that the method sorts the contents of its parameter a.

method InsertionSort(a: array<int>)

requires a != null;

modifies a;

ensures sorted(a) && perm(a[..], old(a[..]) );

{

var i := 1;

while (i < a.Length)

invariant ????

{

var temp := a[i];

var j := i;

while (j > 0 && temp <= a[j - 1])

invariant ????

{

a[j] := a[j - 1];

j := j - 1;

}

a[j] := temp;

i := i + 1;

}

}

Develop a Dafny proof that the method satisfies its specification.

Hints:

* First prove that at the end of the method body, the array a is sorted (ie that sorted(a))
* Now prove that at the end of the method body, the array a is permutation of the original (ie that perm(a[..], old(a[..]) ).

To do this, you can

* (Easier) Prove that each iteration of the outer loop preserves the assertion perm(a[..], old(a[..]).
* (Harder) Prove that each iteration of the inner loop preserves the assertion that the array a where the contents of index j-1 have been replaced by temp is a permutation of the original contents of a.

### **Sample Answers**

Sample answers can be found in file Arrays\_insertion\_Sort\_SA.dfy.