Exercises 6: Loops and Specification Only Fields

TECHNISCHE UNIVERSITÄT DARMSTADT

The solutions to the exercises will be discussed on Monday, 29th June.

Problem 1

In file `Search.java` you find an implementation of the following search algorithm:

Given a 2-dimensional array where the values along any row or any column are *non-decreasing*, and given a value `value` in the array, find an index `x`,`y` such that `value` == `array[x,y]`.

The runtime is linear in the array dimension, i.e. $O(max(n,m))$ for an array of dimension n*m. Therefore the implementation starts in a corner with one index `x` set to 0 and the other `y` set to maximum. If the value at the current position is larger then the required value, increase `x`; if it less, decrease `y`. Return if the value has been found.

a) Specify the method `search` in JML.

b) Provide a loop invariant strong enough to prove the method contract.

c) Try to verify the method.

Problem 2

Consider the class `ArraySanitzer.java` as implemented in `ArraySanitzer.java`. The method `removeDup` removes duplicates from a list of numbers. It receives an int-array `a` and returns a freshly created array containing the same values as `a`, but every value at most once. It is ensured that the method does not modify the entries of array `a`.

- Give a JML postcondition for the method `removeDup` saying that the result array is at most as long as `a`.
- Give a JML postcondition for `removeDup` saying that the result array has no duplicates.
- Give a JML postcondition for `removeDup` saying that every value occurring in `a` also occurs in the result array.
- Give a JML loop specification strong enough to prove the method contract.
- Verify the correctness of method `removeDup`.

Problem 3

The class `Worker`'s `run` method tries to execute the method `doWork` up-to three times.

Upon successful execution of `doWork` (i.e., `doWork` returns true), method `run` terminates and returns `true`. If all three tries fail, false is returned.

- Specify method `run` expressing that in case of success (true is returned). The number of successful tries is exactly one and the number of unsuccessful tries is less than or equal to 2.
- Extend the above specification of method `run` to express that in case that false is returned, none of the three tries was successful.
- Provide a strong enough loop specification.
- Try to verify that method `run` satisfies your specification.

Hint: You might need specification only fields to express above properties.

Problem 4

The interface `StepCounter` as known from previous lectures contains a specification using queries. Change the specification to use model fields instead.

The classes `SimpleStepCounter` and `HistoryStepCounter` contain implementations of `StepCounter`. Add the necessary means to connect specifications and implementations.

Hint: The JML expression (**sum** `int` $i; low <= i$ && $i < high; e$) where e an refer to i can be used to express $\sum_{i=low}^{high-1} e$

Why are model fields better suited than ghost fields in this situation?