Formal Methods in Software Development
Exercise 3 (November 21)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

October 21, 2011

The result is to be submitted by the deadline stated above *via the Moodle interface* of the course as a *.zip or .tgz* file which contains

1. a PDF file with
   - a cover page with the course title, your name, Matrikelnummer, and email address,
   - the deliverables requested in the description of the exercise,
   - a (nicely formatted) copy of the ProofNavigator file used in the exercise,
   - for each proof of a formula F, a readable screenshot of the RISC ProofNavigator after executing the command `proof` F,
   - an explicit statement whether the proof succeeded,
   - optionally any explanations or comments you would like to make;
2. the RISC ProofNavigator (.pn) file(s) used in the exercise;
3. the proof directories generated by the RISC ProofNavigator.

Email submissions are *not* accepted.

Exercise 3: Finding Elements in an Array

Let a be an array of length n of some kind of values and let x be such a value. The problem is to write into (the initial part) of a natural number array p of length n all positions where x occurs in a (in ascending order) and to set the natural number variable m to the number of these occurrences.

For instance, for $a = [2, 3, 2, 3, 5, 7, 2, 2, 9]$, $n = 9$, $x = 2$, we want $p = [0, 2, 6, 7, \ldots]$ and $m = 4$ (by $\ldots$ we denote those values of p that are not of interest).

1. (25P) Give a formal specification of the problem by a pair of a pre-condition P and a post-condition Q.

   Please note in the specification that p must describe *all* occurrences of x in a. Please also note that the only program variables that appear in the problem statement are a, n, x, p, m.

   Validate your specification by checking whether it accepts the variable values given in the example above and whether it rejects for this example the output $p = [0, 2, 4, 6, 7, \ldots]$, $p = [0, 2, 7, \ldots]$, $p = [2, 0, 6, 7, \ldots]$, and $p = [0, 2, 6, 6, 7, \ldots]$ (indicate clearly, why an output is accepted or rejected).

2. (10P) The problem is expected to be solved by the following piece of code:

   ```
   i := 0
   j := 0
   while i < n do
     if a[i] = x then
       b[j] := i
       j := j+1
     end
     i := i+1
   end
   ```

   First assume that you are given a suitable loop invariant I and termination term T. Using these, derive those conditions that have to pe proved to verify the *total* correctness of $\{P\}c\{Q\}$ (you need *not* insert the definitions of P, Q, just use the notation $R[e/x]$ to denote phrase R with variable x substituted by term e).

3. (20P) Now give a suitable definition of I and T. The invariant must apparently express the information available about all positions of a that have already been processed (i.e. the positions less than i) as well as the information about all positions that have not yet been processed (i.e. the positions greater than or equal i). Validate I and T by constructing variable traces for at least three example inputs.

4. (15P) Formalize the proof obligations as a theory of the RISC ProofNavigator. Please note that in the formulation an access to a position i in some array is only legal if i is less than the length of the array.

5. (30P) Prove the obligations with the RISC ProofNavigator.