Formal Methods in Software Development
Exercise 4 (November 28)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

November 3, 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 *.java*/*.theory* file(s) used in the exercise,
   - for each program method, a screenshot of the "Analysis" view of the RISC ProgramExplorer with the specification/implementation of the method and the (expanded) tree of all (non-optional) tasks generated from the method,
   - for each program method, a screenshot of the corresponding "Semantics" view and an informal interpretation of the method semantics;
   - for each task an explicit statement whether the goal of the task was achieved or not and, if yes, how (fully automatic proof, immediate completion after starting an interactive proof, complete or incomplete interactive proof),
   - for each interactive proof, a screenshot of the corresponding "Verify" view with the proof tree,
   - optionally any explanations or comments you would like to make;
2. the *.java*/*.theory* file(s) used in the exercise,
3. the task directory (*.PETASKS**) generated by the RISC ProgramExplorer.

Email submissions are *not* accepted.

Exercise 4: Replacing Elements in an Array

Use the RISC ProgramExplorer[1] to specify the following program, reason about its behavior, and verify its correctness with respect to the specification:

```
class Exercise4
{
  // replaces all occurrences of 'x' in 'a' by 'y'
  public static void replace(int[] a, int x, int y)
  {
    int i = 0;
    int n = a.length;
    while (i < n)
    {
      if (a[i] == x)
        a[i] = y;
      i = i+1;
    }
  }
}
```

In detail, perform the following tasks:

1. (25P) Specify the method by an appropriate contract (clauses `requires`, `assignable`, and `ensures`).
2. (25P) Annotate the loop with an appropriate invariant and termination term (clauses `invariant` and `decreases`),
3. (25P) Investigate the semantics of the method, in particular the method's state relation respectively termination condition derived from the annotations in order to judge the adequacy of your annotations; give an informal interpretation of the semantics and your detailed explanation whether/why it seems adequate.
4. (25P) Verify all (non-optional) tasks generated from the method. Only few of them should require interactive proofs which can be performed by `scatter`/`decompose`, `split`, `case`, and `auto`/`instantiate`.

Do not forget to specify all neccessary side conditions about the pre/post-state of the array such as its non-null status and its length.

[1] Please be sure to use the latest version 1.02 of the RISC ProgramExplorer which fixes a bug in the simplification of formulas which might otherwise affect the verification tasks.