Formal Methods in Software Development
Assignment 3 (November 23)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

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,
   - a section for each part of the exercise with the requested deliverables and optionally any explanations or comments you would like to make;
2. the RISCAL specification (`.txt`) file(s) used in the exercise.

Email submissions are *not* accepted.

Assignment 3: Specifying and Verifying Procedures

We consider the following two problems:

1. Given an array a of length N, return a new array b that has same length as a and is constructed from a by inserting at some position p some element x, shifting all elements at positions greater equal p one element to the right and dropping the last element. For instance, for $a = [1, 2, 3, 4]$, $p = 2$, $x = 0$, we have $b = [1, 2, 0, 3]$. .
2. Given an array a of length N, return a new array b that has the same length as a and is constructed from a by removing the element at some position p, shifting all elements at positions greater than p one element to the left and inserting at the last position some element x. For instance, for $a = [1, 2, 3, 4]$, $p = 2$, $x = 0$, we have $b = [1, 2, 4, 0]$.

Please note that in both problems p may be 0 or $N - 1$ as well.

You are given in the attached files two RISCAL procedures `insert` and `remove` that solve the problem (you can assume that the procedures are correct, you are not allowed to change their implementation).

For each procedure, perform the following tasks:

1. Specify the procedure by pre-and post-conditions. Validate the specification by executing the procedure for all legal inputs.
2. Validate the specification by checking (for small values of array length N and maximum element size M) the tasks under header "verify specification preconditions" and "validate specification" (make sure to switch off option "Silent" when investigating the output of "Execute specification"). For checking the theorems you may use "Apply SMT solver".
3. Annotate the loop(s) in the procedure by invariants and termination measures. Validate the annotations by running the procedure again, thus demonstrating that they are not too strong.

   Please note for the invariants in loop in `insert` that you have to describe the elements of array b at all positions less than i. Please note for the two loops in `remove` that in the second loop you have to describe also the knowledge derived from the execution of the first loop.
4. Check all verification conditions related to the correctness of the implementation ("verify specification preconditions", "verify implementation preconditions", "is result correct?", "verify iteration and recursion"), thus demonstrating that the annotations are strong enough.
5. Prove all verification conditions related to the correctness of the implementation with both methods SMT and MESON, thus demonstrating the correctness of the procedure for arbitrary values of N and M (which should be possible for both proof methods).

Please note that the entry "Apply SMT Solver" in the pop-up menu of a verification condition only checks the condition for *fixed* values of N and M; to actually prove the condition for *arbitrary* values of N and M, one has to select from the top-level menu "TP" one of the methods "SMT" or "MESON" and then select the entry "Apply Theorem Prover" in the pop-up menu of the condition.

The deliverables of this assignment consist of

1. a nicely formatted copy of the RISCAL specifications (included as text, not as screenshots);
2. for both procedures, screenshots of the RISCAL user interface showing the results of all tasks after checking (should be all blue);
3. explicit statements whether the checks of all verification conditions succeeded, whether their proofs succeeded with method SMT, and whether their proofs succeeded with method MESON.
4. for *one* procedure and *one* subproblem of type *Is invariant preserved* that could in MESON not be already solved by automatic decomposition (i.e., the decomposition generated at least one open subproblem) but required proof search, a reasonably detailed explanation of the derivation of this subproblem and its proof (with both options "SMT: Med" and "SMT: Max") illustrated by some screenshots (pop-up menu entry "Open Theorem Prover GUI").