Formal Methods in Software Development
Assignment 1 (November 3)

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 1: Validating and Checking Program Specifications

We consider the following two problems:

1. Given an array a of integers less than or equal m and a positive integer n, find the minimum of the first n elements of a, i.e., that element e that occurs at some of the first n positions of a and that is less than or equal all elements at these positions.
2. Given an array a of integers less than or equal m and a positive integer n, find a position of the minimum of the first n elements of a, i.e., a non-negative integer p less than n such that the element of a at p is that minimum.

In the RISCAL specification file `minimum.txt` you find two procedures `minimumElement` and `minimumPosition` that are supposed to solve these problems, respectively (but one procedure may contain a bug). The specification is based on two integer types `int` and `elem` that bound the domain of possible array indices and elements by constants N and M, respectively.

For *each* of the procedures perform the following tasks (the definitions already given in the specification file must *not* be modified unless explicitly noted):

(a) Formalize the procedure's precondition and postcondition as predicates.

Here do not use the arithmetic quantifier `min` but only the predicate logic quantifiers ∀ and ∃ (translate above specification from natural language to logic). Hint: a formula $(\forall v{:}T$ `with` $F.\ G)$ is equivalent to $(\forall v{:}T.\ F \Rightarrow G)$ and a formula $(\exists v{:}T$ `with` $F.\ G)$ is equivalent to $(\exists v{:}T.\ F \wedge G)$; you may use either notation.

(b) Use the precondition and postcondition to implicitly define a function and check whether the computed results are as desired.

(c) Formulate a theorem that states that some input satisfies the precondition and check this theorem.

(d) Formulate a theorem that states that not every input satisfies the precondition and check this theorem.

(e) Formulate a theorem that states that, for every input that satisfies the precondition, there exists some output that satisfies the postcondition, and check that theorem.

(f) Formulate a theorem that states that, for every input that satisfies the precondition, not every output satisfies the postcondition, and check that theorem.

(g) Formulate a theorem that states that, for every input that satisfies the precondition, the output is uniquely defined by the postcondition, and check that theorem.

(h) Annotate the procedure with preconditions and postconditions and check the correctness of the procedure for every possible array a and integer n; if a condition is a conjunction, it is recommended to use multiple annotation clauses (`requires` and/or `ensures`).

Perform all checks with some small $N \geq 4$ and $M \geq 2$.

Perform the function execution (b) with translation option "Nondeterminism" selected and display option "Silent" *not* selected; all other checks may be performed with option "Nondeterminism" not selected but option "Silent" selected.

If a theorem is not valid, give an explanation of why the theorem is not valid and whether this indicates an error in your specification or not. For this purpose, select the visualization option "Tree" and investigate the evaluation of the theorem (if the tree layout is unsatisfactory, choose larger visualization width/heigth and/or smaller values of N and M).

If the execution of a procedure gives an error, give an explanation of why this is the case and whether this indicates an error in the procedure or not. To investigate the error, you may use the procedure `main` to call the procedure with the input that exhibits the error and use the visualization option "Trace" (alternatively, you may annotate the body of the procedure body with `print` statements). If the error indicates a bug in the procedure, fix this bug.

The deliverables for this exercise consists of the following items:

1. a nicely formatted copy of the extended specification (included as text, not as a screenshot);
2. for each check, a selection of the output of reasonable size (included as text, not as screenshots);
3. if the check of a theorem gives an error, a screenshot of the visualization exhibiting that error, an explanation of the error, and an explanation whether this indicates an error in your specification;
4. if the check of a procedure gives an error, the output that exhibits that error, an explanation of the error, and an explanation whether this indicates an error in your specification or procedure;
5. if you fixed a bug in a procedure, a clear indication and explanation of that fix.