Formal Methods in Software Development
Sample Exam

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

January 7, 2022

Last Name:

First Name:

Matrikelnummer:

Studienkennzahl:

100 points total

1. (25P)

   a) (12P) Write a RISCAL specification (pre/post-condition) of a procedure

   ```
   val N:Nat; type int = Int[-N,N]; type array = Array[N,int];
   proc fill(a:array, p:int, n:int, e:int): array { ... }
   ```

   which returns a copy of *a* where, starting from position *p*, *n* elements have been set to *e*; do not forget to specify suitable preconditions for *p* and *n* that restrict their range to reasonable limits.

   If you prefer, you may also write the specification in the syntax of the RISC ProgramExplorer for a corresponding static Java method.

   b) (13P) Write a *heavy-weight* JML specification for the following method of the Java library (the specification shall be as expressive as possible).

   ```
   public static void fill(int[] a, int fromIndex, int toIndex, int val)

       Assigns the specified int value to each element of the specified range
       of the specified array of ints. The range to be filled extends from index
       fromIndex, inclusive, to index toIndex, exclusive. (If fromIndex==toIndex,
       the range to be filled is empty.)

       Parameters:
           a - the array to be filled
           fromIndex - the index of the first element (inclusive) to be filled
                       with the specified value
           toIndex - the index of the last element (exclusive) to be filled
                     with the specified value
           val - the value to be stored in all elements of the array
       Throws:
           IllegalArgumentException - if fromIndex > toIndex
           ArrayIndexOutOfBoundsException - if fromIndex < 0 or toIndex > a.length
   ```

2. (25P)

   a) (13P) Derive the strongest postcondition of the command c

```
if (i < 10) {
  a[i] = a[i]+3;
  i = i+1;
}
```

   for precondition $a[2] = 5$ (ignoring ‘index out ouf bound” violations). Simplify the derived postcondition as far as possible.

   b) (12P) Derive for above command a judgement of form $c : [F]^{x,\ldots}$ for some state relation F and variable frame $\{x, \ldots\}$.

Remember (for both parts) that an array assignment $a[i] := b$ is just an abbreviation for the scalar assignment $a := a[i \mapsto b]$.

3. (25P) Take the following program which is supposed to compute for given $n \in \mathbb{N}$ the result $s := n^2$:

$\{n = oldn\}$

```
s = 0; i = 1;
while (i <= n)
{
  s = s+2*i-1;
  i = i+1;
}
```

$\{s = n^2 \wedge n = oldn\}$

   a) (13P) Assume you are given a suitable loop invariant I and termination term T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

   b) (12P) Construct for input $n = 10$ a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I it can be concluded that the invariant is preserved (sketch the proof of the corresponding verification condition).

4. (25P) Take the following asynchronous composition of two processes operating on shared variables x, y, i, j where the first process cycles among program counters $P_1 \to P_2 \to P_1 \to \ldots$ and the second process among counters $Q_1 \to Q_2 \to Q_3 \to Q_1 \to \ldots$.

```
initially x = y = i = 0, j = 1
loop                ||   loop
  P1: x = x+j;      ||     Q1: wait i > 0;
  P2: i = 1-i;      ||     Q2: y = y+i;
                    ||     Q3: j = 1-j;
```

a) (10P) Give a formal model of the system (using the interleaving assumption for asynchronous composition) as a *labeled* transition system with five transitions labeled P1, P2, Q1, Q2, and Q3; do not forget the definition of the state space.

b) (6P) Formalize in LTL the properties

- "i becomes greater than zero before y becomes greater than zero (which is eventually the case)"
- "if at any time i becomes greater than zero, then eventually also y will become greater than zero".

c) (9P) Is the second property true for above system? If yes, explain why. If not, show an execution trace that violates this property.

   In the second case, does the property become true, if we assume weak fairness for all transitions? Does it become true, if we assume strong fairness for all transitions? Explain your answers.