Software Testing & Verification 2015/2016
Universiteit Utrecht

30th June 2016, 11:00 - 13:00, EDUC-MEGARON

Lecturer: Wishnu Prasetya

You are allowed to bring along Appendix-A of the LN.

Part I [4pt (8 × 0.5)]

For each question, choose one correct answer.

1. Which of the following is true?

(a) Under the total correctness interpretation, the specification \{\ast \text{true} \ast\} S \{\ast \text{true} \ast\} is always valid.

(b) Under the total correctness interpretation, \{\ast \text{true} \ast\} S \{\ast \text{true} \ast\} implies that S always terminate.

(c) Under the partial correctness interpretation, the specification \{\ast \text{true} \ast\} S \{\ast \text{false} \ast\} is always valid.

(d) Under the partial correctness interpretation, \{\ast \text{false} \ast\} S \{\ast \text{true} \ast\} implies that S does not terminate.

2. Suppose \{\ast P \ast \} S \{\ast Q \ast\} is valid. Which of the following is true?

(a) \{\ast Q \Rightarrow P \ast \} S \{\ast Q \ast\} is also valid.

(b) If additionally \(R \Rightarrow Q\) is valid, then \{\ast P \ast\} S \{\ast R \ast\} is also valid.

(c) If additionally \(P \Rightarrow R\) is valid, then \{\ast R \ast\} S \{\ast Q \ast\} is also valid.

(d) If additionally \(R \Rightarrow P\) is valid, then \{\ast R \ast\} S \{\ast Q \ast\} is also valid.

3. What is the weakest pre-condition of the following statement with respect to the post-condition \(Q\)?

\[
\{\ast ? \ast\} \text{ if } x=y \text{ then } S \text{ else skip } \{\ast Q\ast\}
\]

(a) \(((x=y) \lor wp S Q) \land ((x\neq y) \lor Q)\)

(b) \(((x=y) \Rightarrow wp S Q) \lor ((x\neq y) \Rightarrow Q)\)

(c) \(((x=y) \Rightarrow wp S Q) \land Q\)

(d) \(((x=y) \land wp S Q) \lor ((x\neq y) \land Q)\)
4. What is the **weakest** pre-condition of the following statement with respect to the given post-condition?

\[ \{ \ast \? \ast \} \quad a[0] := x \quad \{ \ast a[a[0]] = x \ast \} \]

(a) \( a(0 \ \text{repl} \ y)[y] = x \)
(b) \( a[y] = x \)
(c) \( a[a(0 \ \text{repl} \ y)[0]] = x \)
(d) \( a(0 \ \text{repl} \ y)[a[0]] = x \)

5. What is the **weakest** pre-condition of the following statement with respect to the given post-condition?

\[ \{ \ast \? \ast \} \]

\( \{ \text{var} \ m; \ \text{assume} \ (\forall i :: m > a[i]) \ ; \ y := m-1 \ \} \)

\[ \{ \ast y > m \ast \} \]

(a) \( m-1 > m \)
(b) \( (\forall i :: m > a[i] \Rightarrow m-1>m') \)
(c) \( (\forall m' :: (\forall i :: m' > a[i]) \Rightarrow m'-1>m) \)
(d) \( (\forall i :: m' > a[i] \land m'-1>m) \)

6. Consider the loop below; \( f \) is a pure function (so, it has no side effect) of type \( \text{int} \rightarrow \text{int} \).

\[ \text{while } x < y \ \text{do } x := f(x) \]

Which of the properties below implies that the above loop terminates?

(a) \( f(x) > 0 \)
(b) \( x>f(x) \land x>0 \) is invariant
(c) for all \( a \) : \( a > f(a) \)
(d) for all \( a \) : \( a < f(a) \)
7. Consider the function \texttt{val} defined below. Given a list of digits, e.g. as in \texttt{val 0 [3, 4, 5]}, it will calculate the integer value of the digits if they would be the string ”345”. In this case the answer is the integer 345. Notice that the function is tail recursive.

\[
\begin{align*}
\texttt{val v} & [ ] = v \\
\texttt{val v} & (x : z) = \texttt{val} (10*v + x) z
\end{align*}
\]

Below is an imperative implementation of the function. The specification is given.

\[
\begin{align*}
\{ \ast \text{true} \ast \} \\
v & := 0 ; t := s ; \\
\textbf{while } t \neq [] \textbf{ do } \\
\{ v := 10*v + \text{head}(t) ; \\
t := \text{tail}(t) \\
\} \\
\{ \ast v = \text{val} 0 s \ast \}
\end{align*}
\]

Which of the following is a consistent and good enough invariant to prove the correctness of the above specification?

(a) \texttt{val v t} = \texttt{val v s} \\
(b) \texttt{val v t} = \texttt{val 0 s} \\
(c) \texttt{v} = \texttt{val 0 t} \\
(d) \texttt{v} = \texttt{sum}[s_i*10^i \mid 0 \leq i < \text{length}(s)]

8. Consider the loop below; \texttt{i} is of type \texttt{int} and \texttt{isPrime(i)} is a side-effect-free function that checks if \texttt{i} is a prime number.

\[
\begin{align*}
\{ \ast 0 < i < N \ast \} \\
\textbf{while } i < N \land \neg \text{isPrime}(i) \textbf{ do } i := i + 1 \\
\{ \ast (i \neq N) \Rightarrow \text{isPrime}(i) \ast \}
\end{align*}
\]

Which of the predicates below is a correct invariant of the loop, that is enough to prove that the above specification is valid under the \emph{partial correctness} interpretation?

(a) \texttt{0 < i < N} \\
(b) \texttt{0 < i < N \land \neg isPrime(i)} \\
(c) \neg \text{isPrime}(i-1) \lor \text{isPrime}(i) \\
(d) (\forall k : 0 < k < i : \neg \text{isPrime}(k))
Part II [6pt]

1. [4 pt] Loop

Consider the following program and its specification. The program checks if every j-th element of an integer array a does not exceed $10^j$ (10 to the power of $j$).

```
{ * 0 ≤ N * }  // pre-condition

p := 1 ;
k := 0 ;
ok := true ;
while k ≟ N ∧ ok do {
    ok := ok ∧ a[k] ≤ p ;
    p := p*10 ;
    k := k + 1 ;
}

{ * ok = (∀j : 0≤j<N : a[j] ≤ 10^j) * }  // post-condition
```

Give a formal proof that the program satisfies its specification, under partial correctness.

- Please mention what your chosen invariant is.
- Every step in your proof should include a justification (the hint/comment part).
- Steps involving quantifiers should be done in small steps: each step should refer to a proof rule or a theorem in Appendix A. You can additionally use this theorem:
  \[ \vdash P \Rightarrow (P = \text{true}) \]
- You don’t have to show the wp calculation.

2. [1 pt] Program call

Consider the following specification of a program to find the next distinct element in an array.

```
{ ∃ k : k>i ∧ a[k] ≠ a[i] }

Iold := i; findOther(READ a : int[], OUT i : int)

{ * i>Iold ∧ a[i] ≠ a[Iold] * }
```

Consider the statement below, that contains a call to findOther. The correctness requirement is expressed as the pre-condition $P_0$ and post-condition $P_3$.

```
{ * $P_0$ : (∀i : (∃k : i < k ∧ a[i] < a[k])) * }

{ $P_1$ : x := a[2i+1] ; i := 2i+1 ;
$P_2$ : findOther(a,i) }

{ * $P_3$ : a[i] ≠ x * }
```
Calculate intermediate predicates $P_2$ and $P_1$, such that from the corresponding positions they guarantee the final post-condition $P_3$. Use the Black Box reduction rule for program call to calculate $P_2$, and standard $\text{wp}$ calculation for $P_1$.

Note that in the above case, the actual parameters passed to the call are literally the same as the formal parameters. Therefore, there is no need to transform the call.

Just give the answers; you do not have to show the calculation. The specification is valid; you can use this fact as a check in your own calculation.

3. [1 pt, challenging] **Termination.**

Consider this program, with the given pre-condition:

\[
\{ * 0 < x < y * \}
\]

\[
\textbf{while } x<y \textbf{ do } \{ \textbf{if } y-x=1 \textbf{ then } y := y+1 \textbf{ else } y := y-2 \} \]

Prove that if the above program starts from the given pre-condition, it will terminate. We do not care in which state the program would terminate.

You can skip the proofs of Init, IC, and EC, but do mention your chosen invariant (make sure it is a consistent invariant).

Mention your termination metric, then prove the conditions TC1 and TC2. Do mention the resulting $\text{wp}$ in TC1.