Assignment 1 CSSE3100/7100 Reasoning about Programs

Due: 4pm on 1 April, 2025

The aim of this assignment is to consolidate your understanding of the course's material on weakest precondition reasoning. It is worth 15% of your final mark for the course.

Submission instructions: Upload a single Dafny file (A1.dfy) to Gradescope with your solutions to Q1-Q3 formatted as per the formatting instructions below.

Q1 Prove the following justifying each line of your proof with a law from Appendix A of *Programming from Specifications*.

(A ==> B) && (!A ==> B) = B **(2 marks)**

Q2 The following program is the third suggested fix for the Zune bug (discussed in Week 1) that appeared online in 2009. At the time, one commentator said "Trying to verify correctness by pure reason—as opposed to trial-and-error testing—looks almost hopeless." Use weakest precondition reasoning to show that the code below is *partially correct*.

```
method CalculateYear(d: nat) returns (year: nat)
    requires d > 0
    ensures year == Year(d, 1980)
{
    year:= 1980;
    var days := d;
    while days > 365
        invariant days >= 0 && Year(days, year) == Year(d, 1980)
    {
        if IsLeapYear(year) {
            if days > 366 {
                days := days - 366;
                year := year+1;
            } else {
                days := days - 366;
            }
        } else {
            days := days - 365;
            year:= year + 1;
        }

    }
}

where

predicate IsLeapYear(y: nat) {
    y%4 == 0 && (y%100 == 0 ==> y%400==0)
}

and
```

```
ghost function Year(d: nat, y: nat):nat {
    if d > DaysInYear(y) then Year(d - DaysInYear(y), y+1) else y
}

ghost function DaysInYear(y: nat): nat{
    if IsLeapYear(y) then 366 else 365
}
```
(9 marks)

Q3 The original Zune bug was due to the program not terminating. Additionally prove that the above program is *totally correct* using weakest precondition reasoning. You will need to decide on an appropriate termination metric to do this. **(4 marks)**

Formatting instructions

The proofs are to be completed in the template file A1.dfy provided on Blackboard. The template enables checking of the syntax of your predicates to avoid you making typos or misusing predicate operators. It also enables us to (partially) auto-grade your submission. For this latter reason, it is important that you carefully follow the instructions below.

The file includes three methods, one for each question above.

The method Q1 has inputs A and B corresponding to the predicates of Q1, and a single declaration of a local Boolean variable PS (for proof state). The proof should be written as a series of assignments to PS. For example, the proof of Exercise 1.5(c) (shown below on the left) should be written as shown on the right below.

```
X && Y ==> Z
!(X&&Y)||Z          (A.22)
(!X||!Y)||Z         (A.18)
!X || (!Y || Z)     (A.5)
X==>!Y||Z           (A.22)
```

```
PS := X && Y ==> Z;
PS := !(X&&Y)||Z;          // A.22
PS := (!X||!Y)||Z;         // A.18
PS := !X || (!Y || Z);     // A.5
PS := X==>!Y||Z;           // A.22
```

Note that justification in terms of laws from *Programming from Specifications* should be included as comments.

The method CalculateYear includes the code from Q2 with two additional local Boolean variables WP and WP_s. The proof should again be written as a series of assignments to WP and WP_s. WP (for weakest precondition) should be used for all predicates in the proof excepting those where a strengthening has occurred. WP_s should be used only when a strengthening has occurred, i.e., the predicate assigned to WP_s should be *strictly stronger* than the predicate below it. For example, the snippet of the final proof in Week 4's lecture slides (shown on the left below) should be written as shown below on the right.

```
{s == n*(n – 1)/2 && n != 33}
                (strengthen with n!=33)
{s == n*(n – 1)/2}          (arithmetic)
{s + n == n*(n – 1)/2 + n}
 s := s + n;
{s == n*(n – 1)/2 + n}
```

```
WP_s := s == n*(n – 1)/2 && n != 33;
// strengthen with n!=33
WP := s == n*(n – 1)/2;   // arithmetic
WP := s + n == n*(n – 1)/2 + n;
 s := s + n;
WP := s == n*(n – 1)/2 + n;
```

Note that justification must be provided for all predicate simplifications. These need to be sufficient for a peer, e.g., someone else doing this course, to understand the step. If you refer to a function in your justification, make sure you state the property of the function you are using. For basic arithmetic, the justification can just be "arithmetic" (as above). The justification can be on the same line as the predicate or, when this would make the overall line too long, on the line following the predicate. If you use the rule proved in Q1, you may include "Q1" as justification.

You also need to justify why the result of your weakest precondition calculation means the method is partially correct. Add this justification as a comment immediately above the top-most line of your proof.

The method Q3 includes the code from Q2 with a ghost variable m to record the value of the termination metric (similar to the ghost variable d in the lecture slides) and two additional Boolean variables WP and WP_s. The proof is to be written following the rules given above for the method CalculateYear. The proof should begin with just the predicate required for proving termination ($d \succ D$ in the lecture slides) and introduce additional predicates by strengthening if required.

You need to justify why the result of your weakest precondition calculation means the method terminates. Add this justification as a comment above the top-most line of your proof.

Marking

For each question, you will be awarded full marks for a correct proof (formatted as described above) with each step (apart from application of wp rules) sufficiently justified. You will lose marks for incorrect proof steps or missing justification as detailed below.

Q1 You will lose 0.5 marks for each incorrect proof step (capped at 1 marks), and 0.5 marks for each case of insufficient or incorrect justification (capped at 1 marks).

Q2 You will lose 0.5 marks for each incorrect proof step (capped at 6.5 marks), and 0.5 marks for each case of insufficient or incorrect justification (capped at 2.5 marks).

Q3 You will lose 0.5 marks for each incorrect proof step (capped at 3 marks), and 0.5 marks for each case of insufficient or incorrect justification (capped at 1 marks).

For each question, additional marks (up to the total marks for the question) may be taken off for work regarded as having little or no academic merit.

School Policy on Student Misconduct

This assignment is to be completed individually. You are required to read and understand the School Statement on Misconduct, available on the School's website at: https://eecs.uq.edu.au/current-students/student-guidelines/student-conduct?p=1#1

This assessment task evaluates students' abilities, skills and knowledge without the aid of generative Artificial Intelligence (AI) or Machine Translation (MT). Students are advised that the use of AI or MT technologies to develop responses is strictly prohibited and may constitute student misconduct under the Student Code of Conduct.