Assignment 2 CSSE3100/7100 Reasoning about Programs

Due: 4pm on 16 April, 2021

The aim of this assignment is to provide you with experience specifying, verifying and deriving iterative programs.

Instructions: Submit a single pdf file with your solution to questions (a) and (b) to Blackboard by the due date and time.

The Calkin-Wilf tree is a binary tree in which each positive rational number appears exactly once. A breadth-first traversal of the tree gives the following sequence of rational numbers 1/1, 1/2, 2/1, 1/3, 3/2, 2/3, 3/1, 1/4, 4/3, 3/5, 5/2, 2/5, 5/3, 3/4, ... where each denominator is the next numerator. The sequence of numerators (and hence denominators) can be calculated by the following function *fusc*, i.e., *fusc(n)* corresponds to the numerator of the *n*th rational number (and hence the denominator of the *(n-1)*th rational number when *n > 1*) in the sequence above.

fusc(0) = 0	(i)
fusc(1) = 1	(ii)
*fusc(2 * n) = fusc(n)*	(iii)
*fusc(2 * n + 1) = fusc(n) + fusc(n + 1)*	(iv)

For the purpose of this assignment, you can assume that rules (iii) and (iv) hold for any integer *n*. Hence, these rules can be applied at any time in your proofs.

(a) Verify whether or not the following program satisfies <u>total</u> correctness. You should use weakest precondition reasoning and may extend the loop invariant if required. You will need to add a `decreases` clause to prove termination.

```
method ComputeFusc(N: int) returns (b: int)
     requires N >= 0
     ensures b == fusc(N)
{
     b := 0;
     var n, a := N, 1;
     while (n != 0)
          invariant fusc(N) == a * fusc(n) + b * fusc(n + 1)
     {
          if (n % 2 == 0) {
               a := a + b;
               n := n / 2;
          } else {
               b := b + a;
               n := (n - 1) / 2;
          }
     }
}
```

(b) Derive a program that returns the position in the above list of rational numbers with numerator `num` and denominator `den`. Your program must be shown to satisfy partial correctness with respect to the following specification. Your program should call `ComputeFusc` but must not call it more than once with the same argument.

```
method ComputePos(num: int, den: int) returns (n: int)
     requires num > 0 && den > 0
     ensures n > 0 && num == fusc(n) && den == fusc(n + 1)
```

Marking

You will get marks for the application of the appropriate rules for each line of code (do not skip the application of a rule), and for correct and, where necessary, justified simplifications. A breakdown of the marks is given below.

(a)	Weakest precondition proof (without termination) [1]	6 marks
	Termination proof	2 marks
(b)	Derived code	3 marks
	Weakest precondition proof	4 marks

[1] Including explanation of program failure if necessary.

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 Schools website at: http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism