CS615 Practice Assignment 1

- ***This assignment is intended to help you get a better understanding of techniques learnt in class by applying them to actual problems.***
- ***Do not turn in solutions to this assignment. These will not be graded.***
- ***Mutual discussion among students is strongly encouraged. However, it is recommended that you finally solve all the problems yourself. Mutual discussion should only be for purposes of clarifying doubts and concepts.***
- ***You are welcome to approach the TA and the instructor for clarifying doubts. Please send an email asking for an appointment.***

1. Consider the following program (should be familiar to those who took Quiz1), where `n1` and `n2` are variables of type `integer` and `L1` and `L2` are variables of type `boolean`.

```
        L1 := false;
        L2 := false;
label1:
        while (n1 + n2 > 0)
          if (L2 = false)
             if (n1 > 0)
                L1 := true;
                n1 := n1 - 1;
             else
                L1 := false;

          if (L1 = false)
             if (n2 > 0)
                L2 := true;
                n2 := n2 - 1;
             else
                L2 := false;

          L2 := not L2;
          L1 := not L1;
```

   (a) Assume that `n1` and `n2` are non-negative at the beginning of execution of the program. Give the weakest precondition that ensures that whenever control reaches `label1`, i.e. at the beginning of each execution of the `while` loop, we have either `L2 = false` or `L1 = false`.
   Use the Floyd-Naur compositional/stepwise induction approach to justify your answer.

   (b) Is it possible for the `while` loop in the above program to iterate forever (non-termination)?
   - If your answer is in the affirmative, you must give the corresponding precondition and prove that the loop never terminates.

- If your answer is in the negative, you must prove that for all possible preconditions, the loop terminates.

Use the Floyd-Naur approach for your proof in either case.

(c) A careless programmer, in trying to code the above program in C, has forgotten to include the first two assignment statements (`L1 := false; L2 := false`). The buggy program, say P_{buggy}, is then sent to a testing engineer whose job is to determine if the program indeed performs what it is supposed to do. The testing engineer cannot look at the internals of P_{buggy} (it has already been compiled into object code), but can provide different initial values of `L1, L2, n1, n2` and compare the final values of these variables after running P_{buggy} with those obtained from a known correct implementation of the program. Of course, the testing engineer must be careful to give only those initial values to variables that guarantee termination of the program.

Using relational semantics of programs, can you determine if there exist initial values of `L1, L2, n1, n2` that would enable the testing engineer detect the problem in P_{buggy}? Assume that because of other constraints, the testing engineer must set the values of `n1` and `n2` to be at least 1000.