CSC410

EXAM 1

Date: October 19, 2022
weight: 20% of course mark, length: 110 minutes

First Name: —————————————–

Last Name: —————————————–

Student Number: ————————————

Read Before You Start

• Write your name and student number, and please do it so that it matches your record on Quercus.

• This exam is designed so that the answers require no English words. You are welcome to write English sentences, but be warned that they will not be graded.

• The exam may look long, but the questions have very short answers. The generous 110-minute time is there to remove any stress from this process. It is not an indication that this exam requires 2 hours of work!

• This exam contains 4 problems (some with multiple subproblems).
Program Correctness Problems

Problem 1: Simple Iterative Correctness

(20 points) Consider the Dafny method below.

```dsharp
method Game(m: nat, n: nat) returns (win: bool)
    requires m == n
    ensures win
{
    var turn := true;
    var a, b := m, n;
    while a > 0 || b > 0
        decreases a, b
        invariant
        { turn <=> (a == b)
            if a > b {
                a := b;
            } else if b > a {
                b := a;
            } else {
                var delta :| delta > 0 && delta <= a;
                a := a - delta;
            }
            turn := !turn;
        }
    win := turn;
    return;
}
```

The postcondition is already specified. Provide the answers to the two items below and stick to Dafny syntax for your answers:

(a) (10 points) Provide the appropriate precondition, a constraint on the inputs m and n, that ensures the postcondition will always be true. Be as liberal as possible, permitting the largest set of pairs of inputs (m,n) for which the postcondition will be satisfied.

(b) (10 points) Provide the appropriate loop invariant that will prove the postcondition as stated, given the precondition that you added for part (a).

The answers are very short. Feel free to write each in front of the question mark in the code, or under each item above.
**Problem 2: Termination**

(10 points) Consider the Dafny method below.

```daffny
method termination(a: nat, b: nat)
{
    var x, y := a, b;
    while y > 0
        decreases ? \( x \mod 3, y \)
        {
            if x \% 3 > 0 {
                x, y := x - 1, y + 1;
            } else {
                x, y := x + 3, y - 1;
            }
        }
}
```

Provide the correct *decreases* clause (termination measure) that proves the loop terminating in Dafny syntax.

Reminder: \( x \mod 3 \) denotes the natural remainder of dividing \( x \) by 3. For example, we have \( 5 \mod 3 = 2 \) and \( 9 \mod 3 = 0 \).
Dataflow Analysis Problems

Problem 3: Possibly Uninitialized Variables

(25 points) Our target is the dataflow analysis that determines if a program variable is possibly uninitialized at a given program point. Here is a precise definition:

A variable \( v \) is possibly uninitialized at program location \( \ell \) if there exists a path from the initial location of the program to location \( \ell \) along which \( v \) is never initialized.

By giving precise answers to the following questions, give a definition to this dataflow analysis. We use \( PU \) (for possibly uninitialize) to denote our sets. If you are using generic sets (corresponding to information about the program) in your definitions, clearly state on the side what these sets are.

- Define the set of dataflow facts.
  \[ D = Vars : \text{the set of program variables} \]

- Is the analysis forward or backward? (circle one)
  
  \[ \text{Forward} \]

- Define the semi-lattice binary operation:
  \[ \forall x, y \in D : x \cap y = \emptyset \]

- Define the transfer functions:
  \[ PU_{exit}(\ell) = PU_{entry}(\ell) - \text{writes}(\ell) \]
  \[ PU_{entry} = \bigcup_{\ell' \rightarrow \ell} PU_{exit}(\ell') \]

- Specify the initialization information by specifying:
  - Initial location (circle one):
    \[ \text{Entry} \]
  
  - Initial value at the location you selected above: \( Vars \)
  
  - Initial value for all other sets: \( \emptyset \)
Problem 4: Lattices

(20 points) Recall that
\[ \forall x, y : x \sqsubseteq y \iff x \sqcap y = x \]
and that \( \sqcap \) we proved in class that \( \sqcap \) denotes the greatest lower bound of \( x \) and \( y \). Show the the following two definitions of monotonicity of a given function \( f \) are equivalent by proving the implications in both directions:
\[ \forall x, y : x \sqsubseteq y \implies f(x) \sqsubseteq f(y) \iff \forall x, y : f(x \sqcap y) \sqsubseteq f(x) \sqcap f(y) \]
Therefore, you will prove the following two statements. Feel free to continue writing on the back of this page if you run out of space here. But, you should not need it. The proofs are very short.

(a) \( \forall x, y : x \sqsubseteq y \implies f(x) \sqsubseteq f(y) \implies \forall x, y : f(x \sqcap y) \sqsubseteq f(x) \sqcap f(y) \)

\begin{align*}
\text{by def.} & \quad \forall x, y \in x \quad \forall x, y \in y \\
& \implies f(x \sqcap y) \in f(x) \sqcap f(y) \\
& \implies f(x \sqcap y) \in f(x) \sqcap f(y) \\
& \implies f(x) \sqsubseteq f(y)
\end{align*}

Since \( \sqcap \) is the greatest lower bound

(b) \( \forall x, y : f(x \sqcap y) \sqsubseteq f(x) \sqcap f(y) \implies \forall x, y : x \sqsubseteq y \implies f(x) \sqsubseteq f(y) \)

\[ x \sqsubseteq y \implies \forall x, y \in x \]

\[ f(x) = f(x \sqcap y) \sqsubseteq f(x) \sqcap f(y) \implies f(x) \sqsubseteq f(y) \]