SENG 2011

Assignment 1

Predicate and Hoare Logic and Program Verification

Due Sunday 9pm, 16th October, 2022

Welcome to the first assignment. Some notes:

- The total number of marks of the exercises will be scaled to 15 marks for the course assessment.
- Exercises 1 and 2 require textual solutions in *pdf* format. Use whatever word processor you like. You may also do them by hand and make an image, but make sure your handwriting is easily readable. You are asked to submit the file *ex12.pdf*.
- Exercises 3, 4, 5 and 6 are Dafny exercises. Submit the individual Dafny program in each case.
- Please make sure your Dafny code verifies with the CSE *dafny* verifier (version 3.7.1).
- None of the exercises requires you to compile code, or generate output.
- Assessment of the Dafny exercises takes into account efficiency, readability, conciseness and structure. (Performance should not be a factor in these exercises.)
- Be careful that you use the given method and file names exactly. The auto-marker is case-sensitive.
- Do not use the *assume* statement, function methods or predicate methods in this assignment.

---

ex1 6 marks

Consider the following statements.

> *If a news report is 'fake news' then the news presenter knows it is incorrect. But sometimes a presenter will make a news report and not know the news is incorrect. So the conclusion is that not all incorrect news is 'fake news'.*

Write predicate formulae that formalises the statements and prove the conclusion correct. Justify each step in the proof.

Submit the solution to this exercise in the file *ex12.pdf*

ex2 6 marks

i For each of the following triples, find a predicate that replaces the question mark to make the Hoare triple correct. Make the predicate as precise as possible.

(a) $\{0 \leq x < 42\} \quad x := x + 1 \quad \{ ? \}$
(b) $\{0 \leq x < 42\} \quad x := 2 * x \quad \{ ? \}$
(c) $\{-42 \leq x < 0\} \quad x := 1 - x \quad \{ ? \}$
(d) $\{0 \leq x \leq y < 42\} \quad y := y - x \quad \{ ? \}$

ii For each of the following triples, replace the question mark with the weakest precondition. Express the precondition in its simplest form.

(a) $\{ ? \} \quad x := 42 \quad \{x := 42\}$
(b) $\{ ? \} \quad x := x + 3 \quad \{x \text{ is even}\}$
(c) $\{ ? \} \quad x, y := 2 * x, x + y \quad \{0 \leq x \leq 42 \ \wedge \ y \leq x\}$
(d) $\{ ? \} \quad x := 2 * y \quad \{42 \leq x \leq y\}$
(e) $\{ ? \} \quad x := 42; \ y := 99 \quad \{x < 99 \ \wedge \ y \leq z\}$

(f) $\{\ ?\ \}$ $x := x + 1;\ y := 2 * x$ $\{y - x = 3\}$
(g) $\{\ ?\ \}$ **if** $x < 50\ \{\ y := 58;\ \}$ **else** $\{\ y := 1;\ \}$ $\{x + y = 100\}$
(h) $\{\ ?\ \}$ **if** $x < 42\ \{$ **if** $x < 100\ \{y := 1;\}$ **else** $\{y := 42;\}\}$ **else** $\{\ y := 2;\ \}$ $\{y$ is even$\}$

Submit the solutions to this exercise in the file *ex12.pdf*

ex3.dfy 6 (method) + 3 (tester) marks

A Palindrome is a word that is the same when written forwards and when written backwards. For example, the word "refer" is a Palindrome.

The Dafny program `ex3errors.dfy`, which you will find on the website, contains a method `PalVerify` that is supposed to verify whether a word is a Palindrome, where the word is represented as an array of characters. The method was written by a novice software engineer, and contains many errors.

i Without changing the signature or the code in the while loop, fix the method so that it veriifes the code. Do not add any Dafny predicates or functions: keep the changes to a minimum.

ii Write a tester method (you may call it anything you like) that verifies that the testcases *refer*, *z* and the empty string are Palindromes, and *xy* and *123421* are not. The tester should not generate any output.

Submit your solution **ex3.dfy**, which should contain:

- method `PalVerify`
- your tester method

ex4.dfy 6 (methods) + 3 (tester) marks

i Write a verified method with signature

```
method Forbid42(x:int, y:int) returns (z: int)
```

that returns $x/(42 - y)$. The method is not defined for $y = 42$.

ii Write a verified method with signature

```
method Allow42(x:int, y:int) returns (z: int, err:bool)
```

If y is not equal to 42 then $z = x/(42 - y)$, otherwise $z = 0$.
The variable *err* is true if $y == 42$, otherwise it is false.

iii Test your two methods by writing a tester with the following testcases. You may call your tester anything you like.

`Forbid42`:

x	y	c
0	1	0
10	32	1
-100	38	-25

`Allow42`:

x	y	c err
0	42	0 true
-10	42	0 true
0	1	0 false
10	32	1 false
-100	38	-25 false

Submit the file **ex4.dfy**, which should contain:

- method `Forbid42`
- method `Allow42`
- your tester method

ex5.dfy 6 marks

Hofstadter *Female* and *Male* sequences can be expressed as:

$$Female(0) = 1 \ ; \ Male(0) = 0$$
$$Female(n) = n - Male(Female(n-1)), \quad n > 0$$
$$Male(n) = n - Female(Male(n-1)), \quad n > 0$$

Write recursive functions in Dafny to represent these sequences.

Submit the file **ex5.dfy**, which should contain:

- function `Female`
- function `Male`

ex6.dfy 8 (predicates/method) + 6 (testing) marks

An array is said to be *clumped* if all equal elements in the array are consecutive in the array. For example, the following arrays are clumped: [42,42,42,1,1], [42], [1,2,-1,8,0]. The following arrays are not clumped: [42,1,42],[0,-1,-1,-1,0]

i Using only 'forall' quantifiers (1 or more), write a Dafny predicate `clumped1` with signature:

```
predicate clumped1(a: array<int>)
```

which returns true if the array a of integers is clumped, and false otherwise.

ii Write a tester method (call it anything you like) that checks the predicate `Clumped1` for a full range of testcases. 'Full' means that boundary cases are checked as well as likely cases such as all elements are equal, all elements are different etc.

iii Using only 'exists' quantifiers (1 or more), write a Dafny predicate `clumped2` with signature:

```
predicate clumped2(a: array<int>)
```

Make sure that this predicate passes the same testcases as part ii, but you need not submit these testcases again.

iv Write a tester method called `ClumpedEquality` that shows predicates `clumped1` and `clumped2` are equivalent for all possible integer arrays.

Submit the file **ex6.dfy**, which should contain:

- predicate `clumped1`
- your tester for `clumped1`
- predicate `clumped2`
- method `ClumpedEquality`

In total, you should submit the files `ex12.pdf`, `ex3.dfy`, `ex4.dfy`, `ex5.dfy` and `ex6.dfy` on the course website. If you have made no attempt at a question, please submit an empty file for that question.

Good luck.