SENG 2011

Assignment 1

Predicate and Hoare Logic and Program Verification

Due Sunday 9pm, 17th October, 2021

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.
- In the Dafny exercises,
  - Please make sure your code verifies with the CSE *dafny* verifier.
  - None of the exercises requires you to compile code, or generate output.
  - In general, there is not a single correct spec of a program. Specs may differ in efficiency, readability, conciseness and structure. All can affect the assessment.
  - If performance is a factor, this will be stated.
  - Use the given predicate, method and file names exactly. The auto-marker is case-sensitive.
  - Do not use the *assume* statement or function methods in this course.

ex1 4 marks The domain of the following predicates is the real numbers:

$P(x, y) : x > y$
$Q(x, y) : x \leq y$
$R(x) : x + 42 = 99$
$S(x) : x > 42$

For each of the following statements, give the truth value, and justify in words the truth value.

i $\forall x \exists y P(x, y)$
ii $\exists y \forall x Q(x, y)$
iii $\forall x \forall y (P(x, y) \vee Q(x, y))$
iv $\exists x R(x)$
v $\forall y (\neg S(y))$
vi $(\exists x S(x)) \wedge \neg(\forall x R(x))$
vii $\exists y \forall x (S(y) \wedge Q(x, y))$
viii $\forall x \forall y ((R(x) \wedge S(y)) \Rightarrow Q(x, y))$

Submit the file ex123.pdf, which should contain the solution to this exercise and **ex2** and **ex3**.

ex2 6 marks If P, Q, R and S are logical propositions, prove $((P \Rightarrow (Q \vee R)) \Rightarrow ((\neg Q \vee S) \wedge \neg S)) \Rightarrow \neg Q$ using just the axioms below.

a) $P \vee \textbf{false} \Leftrightarrow P$
b) $P \vee \textbf{true} \Leftrightarrow \textbf{true}$
c) $P \wedge \neg P \Leftrightarrow \textbf{false}$
d) $P \wedge \textbf{false} \Leftrightarrow \textbf{false}$
e) $P \vee \neg P \Leftrightarrow \textbf{true}$
f) $P \wedge Q \Rightarrow P$
g) $P \wedge Q \Rightarrow Q$
h) $P \wedge Q \Leftrightarrow Q \wedge P$
i) $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$
j) $(P \vee Q) \wedge R \Leftrightarrow (P \wedge R) \vee (Q \wedge R)$
k) $P \wedge (Q \wedge R) \Leftrightarrow (P \wedge Q) \wedge R$
l) $P \Rightarrow Q \Leftrightarrow \neg P \vee Q$
m) $\neg(P \Rightarrow Q) \Leftrightarrow P \wedge \neg Q$
n) $P \Rightarrow P \Leftrightarrow \textbf{true}$

- State on every line in the proof the axiom that you have used.
- No other axioms or inferences may be used.
- For example, to prove $(P \vee \textbf{false}) \Leftrightarrow P$ is **true**, a solution is:

  $(P \vee \textbf{false}) \Leftrightarrow P$ [**Premise**]
  $\Leftrightarrow (P \Leftrightarrow P)$ [**a**]
  $\Leftrightarrow \textbf{true}$ [**n**]

Submit the file ex123.pdf, which should contain the solution to this exercise and **ex1** and **ex3**.

ex3 8 marks The following code fragments show the pre- and post-conditions. Prove the code correct using the inference rules from lectures. Indicate the rule that you use in each step.

i $\{\textbf{true}\}$
```
if x>y then m:=x else m:=y;
```
$\{(m \geq x) \wedge (m \geq y)\}$

ii $\{x = 2^n\}$
```
x:=x*2;
n:=n+1;
```
$\{x = 2^n\}$

iii $\{x = 2^n \wedge (n \leq p)\}$
```
while n<p
do
(x:=x*2;
 n:=n+1;)
```
$\{x = 2^p\}$

where you will need to choose an invariant in iii.

Submit the file ex123.pdf, which should contain the solution to this exercise and **ex1** and **ex2**.

ex4.dfy 4 marks By *adding a specification*, consisting of a pre- and post-condition, an invariant and a decreases clause, verify the following program is correct. Do not change, add or remove any code.

```
method Eval(x:int) returns (r:int) // do not change
{                                  // do not change
  var y:int := x;                  // do not change
  var z:int := 0;                  // do not change
  while y>0                        // do not change
  {                                // do not change
     z := z + x;                   // do not change
     y := y - 1;                   // do not change
  }                                // do not change
  return z;                        // do not change
}                                  // do not change
```

Submit the file ex4.dfy, which should contain this method with specification. No other predicates, functions or methods are required or permitted.

ex5.dfy 10 marks

i Write a predicate with signature

```
predicate exist1(a: array<int>, x:int)
```

that returns **true** if the array contains **exactly one instance** of the number x. Otherwise it returns **false**. For example, if the array is [3,3,1,0,0,0] and x=1 then the predicate should return **true**. If x=0, x=3 or x=42 then the predicate returns **false**.

ii Write a predicate with signature

```
predicate exist2(a: array<int>, x:int)
```

that returns **true** if the array contains **two or more instances** of the number x. Otherwise it returns **false**. For example, if the array is [3,3,1,0,0,0] and x=3 or x=0 then the predicate should return **true**. If x=1 or x=42 then the predicate returns **false**.

iii Write a predicate with signature

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

that returns **true** if the **only zeroes** in an input array are **trailing zeroes**. Otherwise it returns **false**. Note that there may be no zeroes in the array, in which case the predicate is 'vacuously' **true**. Examples of calls that return **true** are for the testcases [3,3,1,0,0,0], [0,0], [9,1,3,8,5] and [ ]. For the arrays [1,0,2] and [0,1] the predicate returns **false**.

iv Write one or more 'tester' methods for these predicates that verify the correctness.

- You can call your tester(s) anything you like.
- Make sure you include the important testcases.
- I recommend you write one tester for each predicate, initially at least. At submission time, you may combine all your testcases into a single tester method (it's your choice), but make sure it's still working. For example:

```
var a: array<int> := new int[][1,42,0,0,0,0,0,0,0,0,0]; // a testcase
assert a[0]==1 && a[1]==42 && a[2]==0 && a[3]==0;       // inform Z3
assert  exist1(a, 42);   // exactly one '42'
assert !exist2(a, 42);   // not more than one '42'
assert  tail(a);         // has a tail
```

- When you declare/define an array in the tester, it is generally a good idea to assert the elements of the array immediately before you call the predicate(s). This informs the theorem prover. You often need to assert just a few relevant elements, and not every element.
- The tester should not generate any output.
- There is a total time limit of 10 secs for the tester. It is important to stay within the timelimit.

General advice:

- You should not try to count the number of instances. The predicates are specs, not programs.
- The predicates may call one another.
- It is possible to write Dafny in such a way that it 'confuses' the theorem prover, resulting in assertion violations for clearly correct input (the prover is unable to prove the assertion) or very poor performance. Change your formulation in that case.

Submit the file ex5.dfy, which should contain the tester(s) and the three predicates only. Make sure these predicates are named correctly (lower case only). No other predicates or functions are required or permitted.

ex6.dfy 8 marks Write a method with signature:

```
method Ceiling7(n:nat) returns (k:nat)
```

that computes the largest number k that is a **multiple of 7** and is less than or equal to some given 'ceiling' number n, where $n >= 0$. For example: if n=43 then the largest number that is a multiple of 7 and is less than or equal to 43 is 42. If n=6 then k=0. If n=1000 then k=994.

As well, write a black-box tester method that calls `Ceiling7()` with appropriate testcases.

- Give the tester any name you like. The tester does not need to generate any output.
- For simplicity, 'hard code' the number 7 into your program instead of passing it as an argument.
- The tester should take less than 5 secs in total.

Submit the file `ex6.dfy`, which should contain method `Ceiling7()`, the tester method, and no other methods. You may include your own Dafny functions and predicates.

In total, you should submit the files `ex123.pdf`, `ex4.dfy`, `ex5.dfy` and `ex6.dfy` on the course website. If you have made no attempt at a question, submit an empty file for that question.

Good luck.