SENG 2011

Assignment 2

Algorithms & Data Structures

Due 1am, Saturday 20th October, 2018

- Please include at the top of your program a comment on any 'issues' that your program has. Examples of issues are: the program does not compile/verify, the program times-out, the program does not pass all testcases. Failure to mention an issue will lead to a loss of marks.
- Unlike the first assignment, there are marks for conciseness and readability in both the code and the verification. This includes, in particular, postconditions and invariants.
- Even if you have installed Dafny on your own computer, confirm your solutions are correct by using the online tool **http://rise4fun.com/Dafny**. I will use that platform to test your solutions.
- Do not use Dafny's 'assume' statement.
- You should always make sure you make postconditions and class invariants are as strong as possible.
- The total number of marks you achieve (out of 70) will be scaled for the purposes of course assessment.

Ex1.dfy 5 marks. A class, `Score`, records the highest score of a game. You may assume the score cannot be negative. The class contains a constructor and a method called `NewScore`. There is also a test method called `TestScore` that is external to the class. There are all described below.

Constructor Initialises the score to zero.

NewScore The input parameter of this method is a score. If the input score is (strictly) higher than the current recorded score then it gets recorded. The method returns 2 variables: a Boolean indicating whether the recorded score changed or not, and the current highest score. Note that we are not keeping a history of scores: just the highest score.

TestScore() This method makes a series of calls to `NewScore` with the scores 0, 2, 0, 4, 4, 6 in that order. The values returned by each call should be verified of course.

Implement, verify and test the class.

Ex2.pdf and Ex2.dfy 5 marks

A toggle switch, sometimes called an actuator, is typically a lever or button that opens and closes an electric circuit. If the switch is open, then toggling the switch closes the circuit. If the switch is closed, then toggling the switch opens the circuit. A toggle switch starts in the closed position.

1. Draw a state-transition diagram that represents the behaviour of a toggle switch.
2. What 'bad' behaviours are possible in this system?
3. Write a Dafny program and specification that models a toggle switch and verify the correct operation of the switch.

Ex3.pdf and Ex3.dfy 20 marks

Typically, using a dishwasher involves loading the dishwasher with dirty dishes and then adding the detergent to the dispenser. The dishwasher then washes the dishes clean, using all the detergent in the process. The dishwasher can then be unloaded, ready to take a new load. At the start, the dishwasher is clean: there are no dishes in the dishwasher and there is no detergent in the dispenser.

There are 4 distinct actions above: loading, adding detergent, washing and unloading. There is some flexibility in the ordering of the actions, as shown by the following 5 testcases:

1) The typical testcase is:

```
Load, AddDtgt, Wash, Unload
```

where the names correspond to the 4 actions above.

2) The detergent could have been added first, so another possible ordering is:

```
AddDtgt, Load, Wash, Unload
```

3) If you know that you are going to do another wash, you could add the detergent much earlier:

```
Load, AddDtgt, Wash, AddDtgt, Unload, Load, Wash, Unload
```

4) If the dishes were really dirty, you may also choose to wash the dishes twice before unloading:

```
Load, AddDtgt, Wash, AddDtgt, Wash, Unload
```

5) You may also load the machine a few plates at a time, or add detergent a bit at a time, so these actions may be repeated:

```
Load, Load, Load, AddDtgt, AddDtgt, Wash, Unload
```

An example of 'bad' behaviour that should fail verification is if the user forgets to add detergent to the dispenser (a risk to hygiene). This corresponds to testcase 1 with the action AddDtgt missing. Another example is if the user forgets to even load the machine (a waste of electricity and detergent), which corresponds to testcase 1 with Load missing. These are not the only examples of behaviour that should be rejected by the Dafny verifier.

In summary, you are asked to:

- draw a state-transition diagram that represents the behaviour of the dishwasher.
- write a specification in Dafny that models the behaviour of the dishwasher described above
- include a method that tests the 5 testcases above, or as many of these testcases or similar testcases as you can manage.
- include testcases that lead to verification errors, but comment these testcases out so your program verifies successfully. Add a comment to each 'bad' testcase that says why failure is warranted.

Addendum. If you complete a specification of the dishwasher that conforms to the description above, and you feel the need for a greater challenge, there are lots of features that can be added to your spec. For example, you could specify the number of dirty and clean dishes, including the maximum number of dishes that can be placed in the machine. This would allow small numbers of dishes to be loaded and unloaded. Alternatively or as well, you can think of a basic feature such as an on/off switch, and different programs (fast, normal and extra-thorough wash), pre-wash, post-wash rinse, and set temperature. If you do add features, choose those that enrich the spec by increasing the complexity of 'inter-actions' and behaviours, both good and bad. In essence, you want features that impact the class invariant and give rise to interesting pre- and post-conditions for actions.

Ex4.dfy 10 marks

The Republic of Mauritius was colonised by the Dutch in the 17th century, the French in the 18th century and the British in the 19th and 20th centuries. The country gained independence in 1968. The national flag of the republic consists of red, blue, yellow and green horizontal stripes (from top to bottom).

Write a Mauritian national flag equivalent of the Dutch national flag problem by sorting an unsorted array of the 4 colours of the Mauritian flag into the correct order and verify that the program is correct. Provide a `Main()` test method that prints the colours of unsorted arrays in correct order.

You should use the Dutch national flag program provided in lectures as starting point, which can be found on the website. Note however that the method should not return any variables. Hence the signature of the method should be `FlagSort(flag:array<Colour>)`.

Ex5.dfy 10 marks

In lectures, the verification of a `Quack` data type was shown (Week 7 lectures, third case study). You will observe that the test method (Slide 29) for the quack calls a method called `HeadTail()` that is missing from the lecture notes. This method swaps the data items at the two ends of the queue (so the item at the head goes to the tail, and the tail goes to the head). No arguments are returned.

Implement this method, verifying its correctness using the ghost sequence `shadow`. The verified code for the `Quack` class and its test method `Main()` can be found on the website.

Ex6.dfy 20 marks

Write a verified method `InsertionSortShuffle(a: array<int>)` that implements an insertion sort using a shuffle-based strategy (as against the swap-based insertion sort that was given in the Week 10 lecture). The sort should be carried out *in situ*, which means that the array self is modified in the sort, and that no other array may be used. There is no return variable.

Test your insertion sort using the following method (I've placed this method on the website for your convenience). You are welcome to use a different test method, but if you do so, explain why you find it necessary. The test method calls a predicate `Sorted(a, low, high)`, which is true if elements at index `i: low<=i<high` in the array `a` are sorted, where the range is within array bounds of course, and false otherwise.

```
method Main()
{
   // do not change this code
   var a := new int[][6,2,0,6,3,5,0,4,1,6,0]; // testcase 1
   var msa := multiset(a[..]);

   InsertionSortShuffle(a);
   assert Sorted(a, 0, a.Length);
   var msa' := multiset(a[..]);
   assert msa==msa';

   var b := new int[][8,7];  // testcase 2
   InsertionSortShuffle(b);

   print a[..], b[..];
}
```

Note you are asked to submit 8 files in this assignment, 2 text files (`Ex2.pdf` and `Ex3.pdf`) and 6 Dafny files (`Ex1.dfy` ... `Ex6.dfy`).