SENG 2011

Assignment 1

Specification & Verification

Due 1am, Saturday 25th Aug, 2018

All the exercises in this assignment require you to write Dafny programs and use the Dafny verifier/compiler to verify your programs. Dafny is available on the website **http://rise4fun.com/Dafny**

Note the following:

Cut & paste. Keep a version of your program on your local computer at all times as the Dafny compiler occasionally loses code that is on the screen (particularly if you have severe errors). Edit your program as much as possible on your local computer, and cut&paste the code into Dafny to verify/compile.

Testcases. All submitted programs should verify without error. No output is required in any of the exercises. Testcases are given in some exercises, but you may add more if you feel it necessary to show particular behaviour.

Submission. There are 7 exercises. In each exercise you need to create a **.dfy** file, the name of which is given. Submit all 7 Dafny files on completion. They are not necessarily in order of difficulty (as indicated by the marks), and the number of marks also doesn't necessarily indicate the degree of difficulty.

Final mark. The total number of marks you achieve will be scaled to arrive at your course assessment mark.

Exercises

Ex1.dfy 1 mark. Verify in Dafny that the following predicate is a tautology.

```
(((p or q) implies r) and (r implies s)) implies (~s implies ~p)
```

where p, q, r and s are Booleans, and   is the negation operator.

Ex2.dfy 1 mark. Verify in Dafny that every integer is either even or odd.

Ex3.dfy 3 marks. The following method consists of a simple while statement with a loop variable that starts at 0 and skips to a limit.

```
method Skippy(limit: nat)
{
   var skip := 7;
   var index := 0;
   while index<=limit
   invariant pred1
   { index := index+skip; }
   assert pred2
}
```

Verify it is correct by filling in the predicates `pred1` and `pred2` (which should be as strong as possible).

Ex4.dfy 8 marks. Write a Dafny method, called `IncDec`, that computes the sum of two integers `x` and `y` by using increments (+1) and decrements (-1) only. So, for example:

```
2 + 4 = 2 +1+1+1+1 = 6
19 + (-3) = 19 -1-1-1 = 16
(-5) + (-3) = -5 -1-1-1 = -8
```

Your method should use iteration, and only a single loop, and the following signature:

```
IncDec(x: int, y: int) returns (sum: int)
```

Include a method `Test` that verifies that `IncDec` behaves correctly for the testcases:

```
5 and 15
5 and -15
5 and 0
-5 and 15
-5 and -15
-5 and 0
```

[My solution is 27 lines.]

Ex5.dfy 6 marks. An array is even-odd sorted if the array consists of even-sorted and odd-sorted sub-arrays. An even-sorted sub-array is an array in which all the values at even indexes are sorted. Analogously for an odd-sorted sub-array. An example is:

```
[2,1,4,2,6,3]
```

In this array, the even-sorted sub-array is [2,4,6], the odd-sorted sub-array is [1,2,3] The arrays [1,2], [2,1] and [] are even-odd sorted. The array [1,2,3,1] is not even-odd sorted.

Write a *Dafny predicate* called `EOSorted` that, given an array, is true if the array is even-odd sorted, and false otherwise. Test the predicate behaves correctly by writing a method `Test` that calls the predicate for all the testcases mentioned above.

Ex6.dfy 6 marks. Write a Dafny method `IsClean` that has the signature:

```
method IsClean(a: array<int>, key: int) returns (clean: bool)
```

The method returns true if and only if the array is clean: that is, if there are no instances of the integer `key` in the array `a`. The array is considered unclean, hence clean is false, if there are one or more instances of the key in the array.

Also write a method `Test` that exercises `IsClean` for the following arrays.

- An array consisting of 5 integers: [1,2,2,2,3]. Verify that the method works correctly for the keys 1, 2, 3 and 4.
- An array consisting of 1 integer: [1], for the keys 1 and 2.
- The empty array for key 1.

[My solution is 43 lines.]

Ex7.dfy 10 marks. Write a Dafny method `Just1` that has the following signature;

```
method Just1(a: array<int>, key: int) returns (b: bool)
```

This method should return true if and only if the array contains just one instance of the key. It should return false otherwise.

Write a method `Test` that exercises `Just1` using the integer array [1,3,3,2,0,2,3,3,4]. The keys 0, 1 and 4 appear just once in this array, so `Just1` should return true for these keys, and false for the keys 2 and 3 that appear more than once, and false for the key 5 that does not appear at all. [My solution is 39 lines.]