Formal Specification- Dafny

Exercise 1

In this exercise, we will consider the **Counter** class (see the file *Counter.dfy*) that represents a simple counter.

Task: Add specifications for all the methods in the class (You may define and use any predicate or function).

Methods **inc** and **dec** are used to increment and decrement the counter (i.e. variable value) respectively, method **getValue** returns the actual value of the counter, the constructor **init** is used to initialize Counter objects and variable value is always greater or equal than zero. Method Main is there only for testing purposes.

Exercise 2

Consider the class `LimitedStack`. A `LimitedStack` object represent a Stack data-structure, following the last-in-first-out principle. In addition, it can only hold a maximum number of elements, given by the field capacity. The stack itself is represented by an array of this length. The current top of the stack is located at index top. If the stack is empty, top is -1.

Task: Formally specify and implement the methods listed below:

2.1 Specify a predicate `Valid`, which specifies the properties and values allowed for fields of objects belonging to the `LimitedStack` class. You will need `Valid` in the specifications of the other methods. Valid should specify the following relationships:

- `arr` is not null
- The capacity must be positive, so we can put at least something on the stack.
- The capacity and the length of `arr` must agree.
- `top` should be at least -1 and strictly less than capacity.

Write down two predicates `Empty` and `Full`, which should specify the value of `top` in each situation.

2.2 The `Init` method

Recall that Dafny initialises object by an **Init** method. Complete its specification, making use of the invariant and predicates from part 2.1 as appropriate:

- The initial capacity of the stack, given by the input parameter **c** must be greater than 0. Consequently, the size of the newly created array `arr` should indeed be **c**.
- Newly created stacks are always empty.
- Note that **Init** methods always typically need to state that they will modify this object, as their purpose is to initialise all the fields. It also needs to specify that `arr` is a newly allocated array (this has been done for you already).

2.3 isEmpty, Peek, Push and Pop

Now write down specifications for the methods `isEmpty, Peek, Push` and `Pop`. Then implement them, and make sure the implementation satisfies the specification.

- `isEmpty` should return true if and only if the stack is empty.
- `Peek` should return the top element of the stack, without removing it.
- `Pop` should remove the top element of the stack and return it.
- `Push` should push an element onto the top of the stack, provided that the stack is not already full.

2.4. A second version of Push

Now, also specify and implement a second version of `push`, called `Push2`. This functions differ from the standard implementation of push from part 2.3 in the following ways:

- It can be called whether or not the stack is full.
- If the stack is full, the oldest element should be dropped to make space. Use the helper method `Shift` to do this, without modifying it or its specification.
- If the stack is not full, the previous elements should be kept as usual.
- Your specification must describe those two behaviours and how they depend on the initial state of the stack. You can use predicates inside **old** too.

2.5 Assertions and Testing

Ensure that Dafny can prove that the tests and assertions in the Main method hold. If not, go back and study your specifications and implementations. Remember that the Dafny verifier cannot see the code inside the method bodies (as opposed to functions or predicates), but only specification pre- and post-conditions.