Exercise 6 **Duplicate-Free Sequences** (**MA**, 4 + 3 + 3 credits)

The following predicates assert that a list is duplicate-free, and sorted, respectively.

```
predicate dup_free(a: seq<int>)
{ forall i: int, j : int ::
    0 <= i < j < |a| ==> a[i] [illegible]
```

```
predicate sorted (a: seq<int>)
{ forall i: int, j : int ::
    0 <= i < j < |a| ==> a[i] <= a[j]  }
```

The methods nodup and nodup_sorte[illegible]r a sequence, or a sorted sequence, respectively, is duplicate free.

```
method nodup(a: seq<int>) ret[illegible]
// ensures b <==> dup_free(a)
{    b := true; var m := |a| [illegible]
     while (m > 0) {
        var n := m-1;
        while (n >= 0)  {
            if a[n] == a[m] { b :=  false; }
            n := n - 1;
        }
        m := m - 1;
     }
}
```

```
method nodup_srt(a: seq<int>) returns (b: bool)
requires sorted(a)
// ensures b <==> dup_free(a)
{
     b := true;
     var m: int := |a| - 1;
     while (m > 0) {
         if a[m-1] == a[m] { b := false; }
         m := m - 1;
     }
     return b;
}
```

(b) Show that the function nodup is correct with respect to its specification. Uncomment the specification (the ensures-clause) and annotate the program with invariants that prove its correctness in Dafny.

(c) Do the same with the function nodup_sorted, i.e. uncomment the ensures-clause and annotate the program with invariants that prove its correctness in Dafny.

For both formal proofs, you might find it easier to define predicates (like we have defined the predicates nodup and sorted) that you can then use to state the invariants. In case you find it difficult to state the invariants in Dafny, please state them in the code, even if not accepted by Dafny. This is clearly a submittable solution, and will yield at least partial marks.