Assignment 1

Theodore S. Norvell

5892 Due 2020 Feb 12.

This assignment may be done in groups of 1 or 2 students. Submission will be by D2L; one submission per group. Please put your work for question 0 and question 1 in separate files. Make sure your name is (or names are) included in comments at the top of the file.

We will talk about Dafny and arrays in class. Here is some useful information. You may also wish to read ahead in the notes.

- If arrays are modified in a method, they must be declared in the method's modifies clause.
- If `a` is an array, then `a[..]` is a sequence (seq) of all the values of the items in the array from left to right. Also `a[..k]` is a sequence of the values of the first `k` items, `a[i..k]` is the sequence of the values of the `k-i` items starting with `a[i]`.
- If `s` and `t` are sequences, `s+t` is the concatenation of the two sequences.
- Universal quantification in Dafny looks like this

  ```
  forall j:int ::  i<=j<k ==> a[j]==0
  ```

  This example says all items of the sequence `a[i..k]` are equal to 0. The `==>` operator is implication. Note that Dafny uses left-to-right 3-valued logic (short-circuiting), so it doesn't matter that `a[j]` is undefined for some values of `j` outside the range of interest.
- Existential quantification is similar. E.g.

  ```
  exists j:int ::  i<=j<k && a[j]==0
  ```

- Dafny allows one to define predicates to be used in assertions and contracts. See below and the notes for examples of predicates.
- You can use `x in s` to say that `x` occurs somewhere within sequence `s`.

Q0 [10] Binary search
Create a verified implementation of the binary search algorithm from slide set 3.
You may want to use this predicate definition

```
predicate Sorted( s : seq<int>)
{
   forall i,j : int :: 0 <= i <= j < |s| ==> s[i] <= s[j]
```

```
}
```

Q1 [20] Sorting.

Design a verified method to sort an array in place. Use any sorting algorithm you like except for selection sort. Ensure that your code verifies. In addition to the Sorted predicate above you may want to use the following predicate

```
predicate PermutationOf( s : seq<int>, t : seq<int> )
{
    multiset(s) == multiset(t)
}
```

Comment your code thoroughly to explain the algorithm and the verification.

Advice: Use procedural abstraction. Don't try to do everything in one method; make smaller methods. that do part of the work. For example, if you implement insertion sort, make a separate method to do the insertion; if you implement quicksort, make a separate method to do the partition.

Advice: Draw pictures of your invariants.

Advice: I found that algorithms that only modify the array by swapping two elements to be the easiest to verify.