IMPERATIVE PROGRAMMING 2013

MOISE GABRIEL

QUESTION 3

(a) Pre : a [0 .. N)

Post : f(a) [0 .. N) such that for all i, j in [0 .. N), if i<j then f(a)(i) ≤ f(a)(j)

(b)

```
object QuickSort {
  // For testing
  var a1 = Array (1,2,3,4,5,6,7)  ; var a5 = a1
  var a2 = Array (1,1,1,1,1)  ; var a6 = a2
  var a3 = Array (9,8,7,6,5,4) ; var a7 = a3
  var a4 = Array (10,5,2,3,5,10,5,5,4,2) ; var a8 = a4
  def partition (l: int, r: int, a: Array [int]) : (int, int) = {
     var pivot = a(l)
     // Invariant I : a [l..i) < pivot && a [i..j) = pivot && a [k .. r) > pivot && l ≤ i < j ≤ k ≤ r
&& a [0.. l) = a0 [0.. l) && a [r.. a.size) = a0 [r .. a0 . size) && a.size = a0. size && a [l.. r) is a
permutation of a0 [l.. r)
     var i = l; var j = l+1; var k = r
     while (j < k)
     { if (a(j) == pivot)  j += 1
         else if (a(j) < pivot) {var t = a(i); a(i) = a(j); a(j) = a(t); i += 1; j += 1}
         else {var t = a(j); a(j) = a(k-1); a(k-1) = t ;  k -= 1}
     }
     (i, j)
  }
  // On a sorted list, we will get (i,j) = (l, r) , so we won't sort anything, and since
partition runs in O(N), the time needed by QSort is linear
  def QSort (l: int, r: int, a: Array [int]) : Unit = {
     if ((r-l) > 1)
       { val (i,j) = partition (l, r, a)
         QSort (l, i, a); QSort (j, r, a)
       }

}
//(c)
def QSort2 (a: Array [int]) : Unit = {
    val q= scala. collection. mutable. Queue [(int, int)] ()
    var N = a.size
    q. enqueue ((0,N))
    // Invariant I : q contains the pairs (i,j) such that all the elements from a[i..j) need
to be sorted and put in the correct places in the sorted array later
    // Variant :   x = #a(i) such that a(i) != a. sorted (i) for i in [0.. N)
    while (! q. isEmpty)
    { var (l,r) = q. dequeue
      var (i,j) = partition(l,r,a)
      if (i-l > 1) q. enqueue ((l,i))
      if (r-j > 1) q. enqueue ((j,r))
    }
 }
//Now, for testing:
def eqArray (a: Array [int], b: Array [int]) : Boolean = {
    var eq = true
    if (a.size != b. size) return false
    var N = a. size
    for (i <-0 until N)  if (a(i) != b(i)) eq = false
    eq
}
def main (args: Array [String]) = {
   QSort(0, a1.size, a1) ; QSort (0, a2.size, a2); QSort(0, a3.size, a3); QSort (0, a4. size, a4)
   assert (eqArray (a1, Array(1,2,3,4,5,6,7))); assert ( eqArray (a2, Array (1,1,1,1,1)))
   assert ( eqArray(a3, Array(4,5,6,7,8,9))); assert (eqArray (a4, Array (2,2,3,4,5,5,5,5,10,10)))
   QSort2 (a5) ; QSort2 (a6) ; QSort2 (a7); QSort2 (a8)
   assert (eqArray (a5, Array(1,2,3,4,5,6,7))); assert (eqArray( a6, Array (1,1,1,1,1)))
   assert (eqArray (a7, Array (4,5,6,7,8,9))); assert (eqArray ( a8, Array (2,2,3,4, 5,5,5,5, 10,10)))
 }

}
```