Example OCL Specifications

Problem Statement

Translate the following informal requirement descriptions into OCL specifications. Please invent new things and names wherever needed.

1. The input is an array A of size $n > 0$. The output is *true* if A is sorted non-decreasingly; otherwise, the output is *false*.

2. The input is an array A of size $n > 0$. Assuming that all elements in A are distinct, the output B is an array storing all elements of A in a non-decreasing order.

Suggested OCL Specifications

(with help from Chi-Jian Luo)

We assume that a template with parameters `eleType` and `k` for single-dimension arrays has been defined in UML. The template includes an attribute named `elements` for storing up to `k` elements of type `eleType`. The template also provides operations/queries `sortNonDec` and `isSortedNonDec` for sorting and for checking sortedness, respectively. The class `intArray` is obtained by an instantiation that binds `eleType` to `integer` and `k` to some integer value.

```
context intArray
def: n: Integer = self.elements->size()

context intArray::isSortedNonDec(): Boolean
pre: n > 0
post: if Sequence{1..n}->forAll(i: Integer |
           Sequence{(i+1)..n}->forAll(j: Integer |
              self@pre.elements->at(i) <= self@pre.elements->at(j))} then
         result = true
     else result = false
     endif

context intArray
def: isDistinct(): Boolean =
        Sequence{1..n}->forAll(i: Integer |
           Sequence{(i+1)..n}->forAll(j: Integer |
```

```
              self.elements->at(i) <> self.elements->at(j)))
def: isPerm(B: intArray): Boolean =  -- check permutation relation
        self.isDistinct() and B.isDistinct()
        implies
        Sequence{1..n}->forAll(i: integer |
            Sequence{1..n}->exists(j: integer |
               self.elements->at(i) = B.elements->at(j)))

context intArray::sortNonDec(): intArray
pre: self.isDistinct()
post: result.isSortedNonDec() and result.isPerm(self@pre)
```