Implementation of a list with a circular array

Variables

sequence: array of positions; each position contains an element and an index
first: integer
last: integer
capacity: integer
invariant: if $first \leq last$, then the elements stored in the positions $sequence[first], \ldots, sequence[last - 1]$ are the elements of the list; otherwise, the elements stored in the positions $sequence[first], \ldots, sequence[capacity - 1]$, $sequence[0], \ldots, sequence[last - 1]$ are the elements of the list; the indices stored in the positions correspond to the indices of the array, that is, the index stored in position $sequence[i]$ is i; *capacity* is the capacity of the array *sequence*

Initialization

$first \leftarrow 0$
$last \leftarrow 0$
$capacity \leftarrow$ capacity of the array

Algorithms

length(*begin*, *end*):
  input: indices of array *sequence*
  output: length of the segment of *sequence* from (and including) *begin* upto (and excluding) *end*
return (*capacity* + *end* − *begin*) mod *capacity*

leftOf(*index*):
  input: index of array *sequence*
  output: index of cell to the left of *index*
return (*capacity* + *index* − 1) mod *capacity*

rightOf(*index*):
  input: index of array *sequence*
  output: index of cell to the right of *index*
return (*index* + 1) mod *capacity*

moveLeft(*begin*, *end*):
  input: indices of array *sequence*
  output: move the segment of *sequence* from (and including) *begin* upto (and excluding) *end* one position to the left
index ← *begin*
while *index* ≠ *end* **do**
loop invariant: *sequence*[*begin*], ..., *sequence*[leftOf(*index*)] have been moved one position to the left
    sequence[leftOf(*index*)] ← *sequence*[*index*]
    index of *sequence*[leftOf(*index*)] ← leftOf(*index*)
    index ← rightOf(*index*)

moveRight(*begin*, *end*):
  input: indices of array *sequence*
  output: move the segment of *sequence* from (and including) *begin* upto (and excluding) *end* one position to the right
index ← *end*
while *index* ≠ *begin* **do**
loop invariant: *sequence*[*index*], ..., *sequence*[leftOf(*end*)] have been moved one position to the right
    sequence[*index*] ← *sequence*[leftOf(*index*)]

index of *sequence*[*index*] ← *index*
index ← leftOf(*index*)

size():
output: size of list
return length(*first*, *last*)

isEmpty():
output: list is empty?
return (*first* = *last*)

first():
precondition: list is nonempty
output: first position of list
return *sequence*[*first*]

last():
precondition: list is nonempty
output: last position of list
return *sequence*[leftOf(*last*)]

before(*position*):
precondition: *position* is not first position and *position* is valid[1]
output: position of list before *position*
index ← index of *position*
return *sequence*[leftOf(*index*)]

after(*position*):
precondition: *position* is not last position and *position* is valid
output: position of list after *position*
index ← index of *position*
return *sequence*[rightOf(*index*)]

isFirst(*position*):
precondition: *position* is valid
output: is *position* first position of list?
return (*position* = first())

isLast(*position*):
precondition: *position* is valid
output: is *position* last position of list?
return (*position* = last())

replace(*position*, *element*):
precondition: *position* is valid
postcondition: element at *position* in list has been replaced with *element*
input: *position* element of which is to be replaced with *element*
output: replaced element
element ← element of *position*
element of *position* ← *element*
return *element*

swap(*first*, *second*):
precondition: *first* and *second* are valid
postcondition: elements of *first* and *second* have been swapped
input: positions elements of which are to be swapped
swap elements of *first* and *second*

[1] *position* is valid if it is part of the list

insertFirst(*element*):
  precondition: array is not full[2]
  postcondition: position with *element* has been inserted at the beginning of list
  input: element to be inserted
  output: position of inserted element
first ← leftOf(*first*)
position ← position with *element* and *first*
sequence[*first*] ← *position*
return *position*

insertLast(*element*):
  precondition: array is not full
  postcondition: position with *element* has been inserted at the end of list
  input: element to be inserted
  output: position of inserted element
position ← position with *element* and *last*
sequence[*last*] ← *position*
last ← rightOf(*last*)
return *position*

insertBefore(*position*, *element*):
  precondition: array is not full and *position* is valid
  postcondition: position with *element* has been inserted before *position* in list
  input: *element* to be inserted before *position*
  output: position of inserted element
index ← index of *position*
if length(*first*, *index*) ≤ length(*index*, *last*) **then**
    moveLeft(*first*, *index*)
    temp ← position with *element* and leftOf(*index*)
    sequence[leftOf(*index*)] ← *temp*
    first ← leftOf(*first*)
else
    moveRight(*index*, *last*)
    temp ← position with *element* and *index*
    sequence[*index*] ← *temp*
    last ← rightOf(*last*)
return *temp*

insertAfter(*position*, *element*):
  precondition: array is not full and *position* is valid
  postcondition: position with *element* has been inserted after *position* in list
  input: *element* to be inserted after *position*
  output: position of inserted element
if *position* = last() **then**
    return insertLast(*element*)
else
    return insertBefore(after(*position*), *element*)

remove(*position*):
  precondition: *position* is valid
  postcondition: *position* has been removed from list
  input: position to be removed
  output: element of removed position
element ← element of *position*

[2] *capacity* − size() ≥ 2

```
index ← index of position
if length(first, index) ≤ length(rightOf(index), last) then
    moveRight(first, index)
    first ← rightOf(first)
else
    moveLeft(rightOf(index), last)
    last ← leftOf(last)
return element
```