Implementation of a binary tree with an array

Variables

tree: array of positions; each position contains an element and an index
size: integer
invariant: *tree*$[i]$ contains (e, i) if and only if the binary tree has a node with level numbering i and element e; *size* is the size of the binary tree.

Initialization

size $\leftarrow 0$

Algorithms

size():
  output: size of binary tree
return *size*

isEmpty():
  output: binary tree is empty?
return $(size = 0)$

removeAboveExternal(*position*):
  precondition: *position* is external and different from root
  postcondition: *position* and its parent have been removed from tree and subtree rooted at sibling of *position* takes place of its parent
  input: position of binary tree
  output: element of parent of *position*
index $\leftarrow$ index of *position*
parent $\leftarrow$ *index* div 2
element $\leftarrow$ *tree*[*parent*]
if $index = 2 * parent$ **then**
    sibling $\leftarrow index + 1$
else
    sibling $\leftarrow index - 1$
move(*sibling*)
return *element*

move(*index*):
  precondition: $index > 1$ and *tree*[*index*] contains a position
  postcondition: subtree rooted at *tree*[*index*] has been moved to *tree*[*index* div 2] (removing its parent and sibling)
  input: index of node in binary tree
tree[*index* div 2] $\leftarrow$ *tree*[*index*]
if *tree*$[2 * index]$ contains a position **then**
    move($2 * index$)
    move($2 * index + 1$)