Implementation of a deque with a doubly linked list with dummy nodes

Variables

size: integer
deque: doubly linked list with dummy nodes at the front and the rear; each node, apart from the dummy nodes n_0 and n_{m+1}, contains an element of the deque

$$\begin{tikzcd}
n_0 \arrow[r, bend left] & n_1 \arrow[l, bend left] \arrow[r, bend left] & n_2 \arrow[l, bend left] \arrow[r, bend left] & \cdots \arrow[l, bend left] \arrow[r, bend left] & n_{m-1} \arrow[l, bend left] \arrow[r, bend left] & n_m \arrow[l, bend left] \arrow[r, bend left] & n_{m+1} \arrow[l, bend left]
\end{tikzcd}$$

header: pointer to node
trailer: pointer to node
invariant: the nodes $n_1, \ldots, n_m$ of *deque* contain the elements of the deque listed from front to rear. *size* is the size of the deque. *header* points to n_0 and *trailer* points to n_{m+1}.

Initialization

size $\leftarrow 0$

deque $\leftarrow$

$$\begin{tikzcd}
n_0 \arrow[r, bend left] & n_1 \arrow[l, bend left]
\end{tikzcd}$$

header points to n_0
trailer points to n_1

Algorithms

```
size()
   output: size of deque
return size
```

```
isEmpty()
   output: deque is empty?
return ($size = 0$)
```

```
first()
   precondition: deque is nonempty
   output: element at the front of deque
return element of second node of deque
```

```
last()
   precondition: deque is nonempty
   output: element at the rear of deque
return element of one but last node of deque
```

```
insertFirst(element)
   postcondition: element has been added to the front of deque
   input: element to be added to deque
add new node with element element in between header and the second node of deque
$size \leftarrow size + 1$
```

```
insertLast(element)
   postcondition: element has been added at the rear of deque
   input: element to be added to deque
add new node with element element in between trailer and the one but last node of deque
$size \leftarrow size + 1$
```

```
removeFirst()
   precondition: deque is nonempty
```

```
  postcondition: first element has been removed from deque
  output: first element of deque
temp ← element of second node of deque
remove second node from deque
size ← size − 1
return temp

removeLast()
  precondition: deque is nonempty
  postcondition: last element has been removed from deque
  output: last element of deque
temp ← element of one but last node of deque
remove one but last node from deque
size ← size − 1
return temp
```