|  |  |  |
| --- | --- | --- |
| **Stack ADT** | | |
| Stack =  = stack bottom | | |
| {inv: 0 ≤ n ʌ Size (Stack) = n ʌ top = } | | |
| Operations: | | |
| * Stack (constructor) | - | → Stack |
| * Push (modifier) | Stack x Node | → Stack |
| * Pop (modifier) | Stack | → Stack |
| * Top (modifier) | Stack | → Node |
| * isEmpty (analyzer) | Stack | → Boolean |

|  |
| --- |
| **Stack ( - )** |
| "Builds an empty stack"  {pre: - }  {post: Stack s = } |

|  |
| --- |
| **push (Stack s, Node)** |
| " Adds a new node to stack s"  {pre: Stack s = and a node or s = and a node}  {post: Stack s = or s = { }] |

|  |
| --- |
| **Pop (Stack)** |
| "Extracts from the stack s, the most recently inserted (pushed) element. "  {pre: Stack }  {post: Stack = } |

|  |
| --- |
| **Top (Stack)** |
| “Recovers the value of the element on the top of the stack.”  {pre: Stack }  {post: Node } |

|  |
| --- |
| **isEmpty (Stack)** |
| "Determines if the stack s is empty or not."  {pre: Stack s}  {post: True if s = ∅, False if s ∅} |

|  |  |  |
| --- | --- | --- |
| **Queue ADT** | | |
| Queue = | | |
| {inv: 0 ≤ n ʌ Size (Queue) = n ʌ front = ʌ back = } | | |
| Operations: | | |
| * Queue (constructor) | - | → Queue |
| * Enqueue (modifier) | Queue x Node | → Queue |
| * Dequeue (modifier) | Queue | → Node |
| * Front (modifier) | Queue | → Node |
| * isEmpty (analyzer) | Queue | → Boolean |

|  |
| --- |
| **Queue ( - )** |
| "Builds an empty queue"  {pre: - }  {post: Queue q = } |

|  |
| --- |
| **Enqueue (Queue q, Node)** |
| "Inserts a new node to the back of the queue q"  {pre: Queue q = and a node or q = and a node}  {post: Queue q = or s = { }] |

|  |
| --- |
| **Dequeue (Queue)** |
| "Extracts the element in Queue q’s front "  {pre: Queue }  {post: Stack = and a node } |

|  |
| --- |
| **Front (Stack)** |
| “Recovers the value of the element on the top of the stack.”  {pre: Queue }  {post: Node } |

|  |
| --- |
| **isEmpty (Queue)** |
| "Determines if the Queue q is empty or not. "  {pre: Queue q}  {post: True if q = ∅, False if q ∅} |

|  |  |  |
| --- | --- | --- |
| **Queue ADT** | | |
| Queue = | | |
| {inv: 0 ≤ n ʌ Size (Queue) = n ʌ front = ʌ back = } | | |
| Operations: | | |
| * Queue (constructor) | - | → Queue |
| * Enqueue (modifier) | Queue x Node | → Queue |
| * Dequeue (modifier) | Queue | → Node |
| * Front (modifier) | Queue | → Node |
| * isEmpty (analyzer) | Queue | → Boolean |

|  |
| --- |
| **Queue ( - )** |
| "Builds an empty queue"  {pre: - }  {post: Queue q = } |

|  |
| --- |
| **Enqueue (Queue q, Node)** |
| "Inserts a new node to the back of the queue q"  {pre: Queue q = and a node or q = and a node}  {post: Queue q = or s = { }] |

|  |
| --- |
| **Dequeue (Queue)** |
| "Extracts the element in Queue q’s front "  {pre: Queue }  {post: Stack = and a node } |

|  |
| --- |
| **Front (Stack)** |
| “Recovers the value of the element on the top of the stack.”  {pre: Queue }  {post: Node } |

|  |
| --- |
| **isEmpty (Queue)** |
| "Determines if the Queue q is empty or not. "  {pre: Queue q}  {post: True if q = ∅, False if q ∅} |

|  |  |  |
| --- | --- | --- |
| **Linked List ADT** | | |
| LinkedList =  = first | | |
| { inv: 0 ≤ n ʌ Size (LinkedList) = n } | | |
| Operations: | | |
| * LinkedList (constructor) | - | → LinkedList |
| * add (modifier) | Node | → LinkedList |
| * remove (modifier) | Integer | → LinkedList |
| * isEmpty (analyzer) | LinkedList | → Boolean |
| * size (analyzer) | LinkedList | → Integer |
| * getNode (analyzer) | Integer | → Node |

|  |
| --- |
| **LinkedList ( - )** |
| "Builds an empty linked list"  {pre: - }  {post: LinkedList l = } |

|  |
| --- |
| **Add (Node)** |
| "Add a new node to the linked list"  {pre: LinkedList l = and a node or l = and a node}  {post: LinkedList l = or l = { }] |

|  |
| --- |
| **Remove (Integer i)** |
| "Remove the node at the index i"  {pre: LinkedList }  {post: LinkedList = } |

|  |
| --- |
| **isEmpty ( LinkedList l )** |
| "Determines if the linked list l is empty or not."  {pre: LinkedList l}  {post: True if l = ∅, False if l ∅} |

|  |
| --- |
| **Size (LinkedList)** |
| "Determines the number of elements in the LinkedList."  {pre: LinkedList l}  {post: Size (LinkedList) = n; n ≥ 0} |

|  |
| --- |
| **GetNode (Integer i)** |
| "Returns a node at index i"  {pre: LinkedList }  {post: Node } |