TAD Stack
Stack= {⟨e1,e2,e3,...,en⟩,top }
{inv: 0 ≤ n ∧ Size(Stack) = n ∧ top = en }
Stack: <> ------> <Stack> **Constructor** push: <element> ------> <> **Modifier** pop: <> -----> <en> **Modifier** top: <> -----> <en> **Analyzer** empty: <> -----> <boolean> **Analyzer** getSize: <> -----> <int> **Analyzer** stackToArray: <> -----> <Element[Size]> **Analyzer**

Stack()
Create an empty stack
{pre:TRUE}
{post: Stack s = ∅ }}

pop()
Extracts from the stack s,the most recently inserted element.
{pre:Stack s != ∅ i.e. s=⟨e1,e2,e3,...,en⟩}
{post: Stack s =⟨e1,e2,e3,...,en−1⟩ }

push(element)
Put an item on the stack
{pre: Stack s =⟨e1,e2,e3,...,en⟩and element e or s=∅ and element e }
{post: Stack s =⟨e1,e2,e3,...,en,e⟩or s=⟨e⟩}}

top()
Recovers the value of the element on the top of the stack.

{pre:Stacks !=∅ i.e. s=⟨e1,e2,e3,...,en⟩}
{post: Element en }

empty()
Check if the stack is empty
{pre: Stack s}
{post:True if s = ∅, False if s != ∅ }

getSize()
get the size of the stack
{pre: Stack s =⟨e1,e2,e3,...,en⟩}
{post: n }

stackToArray()
get the array of the stack
{pre: Stack s =⟨e1,e2,e3,...,en⟩}
{post: Element[n]}

TAD HashTable
HashTable = { <(h(k1,v1), h(k2,v2),....h(kn,vn)> }
{inv: H(h(k1,v1), h(k2,v2),....h(kn,vn)|(kn,vn) != null. H table, h element, k key, v value }
HashTable: <> ----------> <HashTable> **Constructor** hashFuntion: <k> ------> <K> **Analyzer** put: <K, V> ------> <HashTable> **Modifier** search: <K> -----> <h> **Analyzer** remove: <K> -----> <h> **Modifier** getElements: <K> -----> <HashTable> **Analyzer** getSize: <HashTable> -----> <h> **Analyzer** bookList: <> -----> <n> **Modifier**

HashTable()
Create an empty HashTable
{pre:TRUE}
{post: HashTable s = ∅}

getElements()
get the elements of the hashTable
{pre: Hash Table s = <(h(k1,v1), h(k2,v2),....h(kn,vn)>}
{post: h[n]}

getSize()
get the size of the hashTable
{pre:<(h(k1,v1), h(k2,v2),....h(kn,vn)> }
{post: n }

search(K)
search a element of the hashTable
{pre: K != null && Hash Table s = <(h(k1,v1), h(k2,v2),....h(kn,vn)>}
{post: V}

remove(K)
remove a element of the hashTable
{pre:{pre: K != null && Hash Table s = <(h(k1,v1), h(k2,v2),....h(kn,vn)>}}
{post:<(h(k1,v1), h(k2,v2),....h(kn-1,vn-1)>}

bookList()
pass the hashTable to a list
{pre: Hash Table s = <(h(k1,v1), h(k2,v2),....h(kn,vn)>}
{post: ArrayList e = <(h(k1,v1), h(k2,v2),....h(kn,vn)>}

put(K,V)
insert an element in the hashTable
{pre: Hash Table s = <(h(k1,v1), h(k2,v2),....h(kn,vn)> }
{post: <(h(k1,v1), h(k2,v2),....h(kn,vn),h(K,V)>}

TAD Queue
Queue= <e1,e2,e3,...,en> front, back>
0 ≤ n ∧ size(Queue) =n ∧ front=e1 ∧ back=en
Queue: < > → < > **Construction** enqueue: < element> → < > **Modifier** dequeue: <>→<e1> **Modifier** front: < > → <e1> **Analyzer** isEmpty: < > → <boolean> **Analyzer**

Queue()
Builds an empty queue
{pre: }
{pos: Queue q= ∅}

enqueue ()
Inserts a new element e to the back of the queue q
{pre: e ϵ e}
{pos: q=<e1,e2,e3,...en,e〉 or q=<e>

dequeue()
Extracts the element in Queueq’s front
{pre: {Queue q ≠ ∅ }
{pos: Queue q=〈e2,e3,e4,...,en−1〉 and e1 }

front()
Recovers the value of the element on the front of the queue.
{pre: {Queue q ≠ ∅ }
{pos: e1}

isEmpty()
Determines if the Queue q is empty or not.
{pre: Queue q }
{pos: True if q=∅, False if q ≠∅}