TAD Stack

Stack= $\{\langle e1, e2, e3, ..., en \rangle, top \}$

 $\{inv: 0 \le n \land Size(Stack) = n \land top = en \}$

 Stack:
 <> -----> < Stack>
 Constructor

 push:
 <= lengty:</td>
 Modifier

 Modifier
 Modifier

 Analyzer

 Analyzer

getSize: <> ----> <int> Analyzer

stackToArray: <> ----> <Element[Size]> Analyzer

Stack()

Create an empty stack

{pre:TRUE}

{post: Stack $s = \emptyset$ }}

pop()

Extracts from the stack s,the most recently inserted element.

{pre:Stack s != \emptyset i.e. s= \langle e1,e2,e3,...,en \rangle }

{post: Stack $s = \langle e1, e2, e3, ..., en-1 \rangle$ }

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 = $\langle e1,e2,e3,...,en,e\rangle$ or s= $\langle e\rangle$ }}

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 $s = \emptyset$, False if $s != \emptyset$ }

```
getSize()
```

get the size of the stack

{pre: Stack $s = \langle e1, e2, e3, ..., en \rangle$ }

{post: n }

stackToArray()

get the array of the stack

{pre: Stack $s = \langle e1, e2, e3, ..., en \rangle$ }

{post: Element[n]}

TAD HashTable

HashTable = $\{ <(h(k1,v1), h(k2,v2),....h(kn,vn) > \}$

 $\{\text{inv: H(h(k1,v1), h(k2,v2),....h(kn,vn)}| \text{kn,vn}\} = \text{null. H table, h element, k key, v value} \}$

 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 = \emptyset }

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>

 $0 \le n \land size(Queue) = n \land front=e1 \land back=en$

Queue: $<> \rightarrow <>$ Construction

enqueue: < element $> \rightarrow <$ > Modifier

dequeue: $<> \rightarrow <$ e1> **Modifier**

front: $<> \rightarrow <e1>$ Analyzer

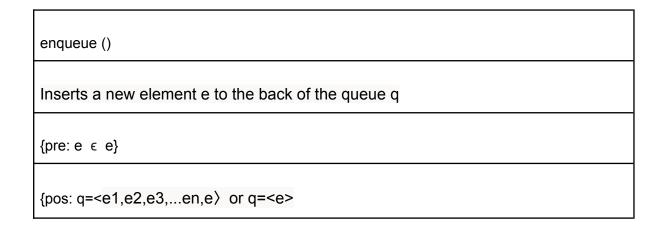
isEmpty: $<> \rightarrow <$ boolean> Analyzer

Queue()

Builds an empty queue

{pre: }

{pos: Queue q= ∅}



```
dequeue()

Extracts the element in Queueq's front

{pre: {Queue q \neq \emptyset}

{pos: Queue q = \langle e2,e3,e4,...,en-1 \rangle and e1 }
```

```
front()

Recovers the value of the element on the front of the queue.

{pre: {Queue q \neq \emptyset}}

{pos: e1}
```

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