| **Hash Table ADT** |
|---|
| $HashTable = \{\, e_1 = \langle key, value\rangle,\; e_2 = \langle key, value\rangle,\; e_n = \langle key, value\rangle \,\}$ |
| $\{\, inv: size(HashTable) < Universe$ $\;\; e_1.key \neq e_2.key \wedge e_1.key \neq e_n.key \wedge e_2.key \neq e_n.key \,\}$ |
| $Operations$: <br> — $HashTable$: $\rightarrow HashTable$ <br> — $hash: key$ $\rightarrow Integer$ <br> — $put: HashTable\ x\ e$ $\rightarrow HashTable$ <br> — $get: HashTable\ x\ key$ $\rightarrow value$ <br> — $remove: HashTable\ x\ key$ $\rightarrow HashTable$ <br> — $isEmpty: HashTable$ $\rightarrow Boolean$ <br> — $size: HashTable$ $\rightarrow Integer$ |

Constructor:

| ***HashTable*** () |
|---|
| "*Generate a new Hash table with the empty data.*" |
| { *pre*: True } |
| { *post*: HashTable<> } |

| ***hash*** (***key***) |
|---|
| "*Generates a number by applying the hash function to the entered key that indicates the position of the element.*" |
| $\{\, pre: HashTable =<> \vee HashTable =< elem_1, elem_2, \ldots\ elem_n > \}$ |
| $\{\, post: hash(key) = position \rightarrow position\ of\ the\ hashTable \,\}$ |

Modifiers:

| ***put*** (***HashTable***, ***element***) |
|---|
| "*Adds an element to the HashTable.*" |
| $\{\, pre: HashTable =<> \vee HashTable =< elem_1, elem_2, \ldots\ elem_n > \}$ |
| $\{\, post: HashTable =< elem_1, elem_2, \ldots\ elem_n, elem_{n+1} > \}$ |

| $\boldsymbol{remove}\ (\boldsymbol{HashTable, key})$ |
|---|
| $"Removes\ an\ element\ from\ the\ Hash\ table."$ |
| $\{\ pre: HashTable =< elem_1, elem_2, \dots\ elem_n >\}$ |
| $\{\ post: key\ \in HashTable \rightarrow HashTable =< elem_1, elem_2, \dots\ elem_{n-1} >$ $\vee\ key\ \notin HashTable \rightarrow False\ \}$ |

Analyzers:

| $\boldsymbol{get}\ (\boldsymbol{HashTable, key})$ |
|---|
| $"Returns\ the\ value\ of\ an\ element\ of\ the\ Hash\ table."$ |
| $\{\ pre: HashTable =< elem_1, elem_2, \dots\ elem_n >\}$ |
| $\{\ post: key\ \in HashTable \rightarrow value\ \vee key\ \notin HashTable \rightarrow False\}$ |

| $\boldsymbol{isEmpty}\ (\boldsymbol{HashTable})$ |
|---|
| $"Evaluates\ whether\ the\ Hash\ table\ is\ empty\ or\ not."$ |
| $\{\ pre: HashTable =<> \vee HashTable =< elem_1, elem_2, \dots\ elem_n >\}$ |
| $\{\ post: HashTable <> \rightarrow true\ \vee HashTable < elem_1, elem_2, \dots\ elem_n > \rightarrow false\}$ |

| $\boldsymbol{size}\ (\boldsymbol{HashTable})$ |
|---|
| $"Returns\ the\ number\ of\ elements\ in\ the\ Hash\ table."$ |
| $\{\ pre: HashTable =<> \vee HashTable =< elem_1, elem_2, \dots\ elem_n >\}$ |
| $\{\ post: size(HashTable) = n, n \rightarrow\ Number\ of\ HashTable\ elements\}$ |

| **Stack ADT** |
|---|
| $Stack\ = \{\ e_1 = \langle element \rangle$ $e_n = \langle element \rangle\ \}$ |
| $\{\ inv: \forall i, j \in 1,2, \dots, n, (i \neq j \Rightarrow ei \neq ej)$ $n \geq 0\ \}$ |
| $Operations$: — $Stack$: $\rightarrow Stack$; — $push: Stack\ x\ element$ $\rightarrow Stack$; — $pop: Stack$ $\rightarrow Stack$; — $top: Stack$ $\rightarrow element$; — $isEmpty: Stack$ $\rightarrow Boolean$ |

Constructor:

| $\boldsymbol{Stack}$ () |
|---|
| "$\mathit{Generates\ a\ new\ empty\ Stack.}$" |
| $\{\ pre: true\ \}$ |
| $\{\ post: Stack <>\}$ |

Modifiers:

| $\boldsymbol{push}$ $(\boldsymbol{Stack}, \boldsymbol{element})$ |
|---|
| "$\mathit{Stores\ the\ new\ element\ at\ the\ end\ of\ the\ Stack.}$" |
| $\{\ pre: Stack <> \vee Stack < \cdots, elem_n >\}$ |
| $\{\ post: Stack < elem_n, elem_{n+1} >\}$ |

| $\boldsymbol{pop}$ $(\boldsymbol{Stack})$ |
|---|
| "$\mathit{Removes\ the\ last\ element\ from\ the\ stack.}$" |
| $\{\ pre: Stack < \cdots, elem_n >\}$ |
| $\{\ post: Stack < \cdots, elem_{n-1} >\}$ |

Analyzers:

| $\boldsymbol{top}$ $(\boldsymbol{Stack})$ |
|---|
| "$\mathit{Returns\ the\ last\ element\ in\ the\ Stack.}$" |
| $\{\ pre: Stack < \cdots, elem_n\}$ |
| $\{\ post: peek(Stack) \rightarrow elem_n\}$ |

| $\boldsymbol{isEmpty}$ $(\boldsymbol{Stack})$ |
|---|
| "$\mathit{Evaluates\ whether\ the\ Stack\ is\ empty\ or\ not.}$" |
| $\{\ pre: Stack <> \vee Stack < \cdots, elem_n >\}$ |
| $\{\ post: Stack <> \rightarrow true\ \vee Stack < \cdots, elem_n > \rightarrow false\ \}$ |

| **PriorityQueue ADT** |
|---|
| $PriorityQueue = \{ (e_1,\ p_1), (e_2,\ p_2), \dots, (e_n,\ p_n) \}$ |
| $\{ inv: e_i \in U \wedge p_i \in \mathbb{N} \wedge$ <br> $\forall i,j \in \{1,\dots,n\}\ con\ i < j \rightarrow p_i \geq p_j \}$ |
| $Operations$: <br><br> — $PriorityQueue$: $\rightarrow PriorityQueue$ <br> — $enqueue: PriorityQueue\ x\ e\ x\ p$ $\rightarrow PriorityQueue$ <br> — $dequeue: PriorityQueue$ $\rightarrow e$ <br> — $peek: PriorityQueue$ $\rightarrow e$ <br> — $increasePriority: PriorityQueue$ $\rightarrow PriorityQueue$ <br> — $prioritizeLowest: PriorityQueue$ $\rightarrow PriorityQueue$ <br> — $isEmpty: PriorityQueue$ $\rightarrow boolean$ <br> — $size: PriorityQueue$ $\rightarrow Integer$ |

Constructor:

| $\boldsymbol{PriorityQueue}$ () |
|---|
| "$Generates\ a\ new\ empty\ PriorityQueue.$" |
| $\{ pre: true \}$ |
| $\{ post: PriorityQueue\{\ \ \} \}$ |

Modifiers:

| $\boldsymbol{enqueue}$ ($\boldsymbol{PriorityQueue, e, p}$) |
|---|
| "$Adds\ an\ element\ and\ its\ respective\ priority\ to\ the\ PriorityQueue.$" |
| $\{ pre: e \in U \wedge p \in \mathbb{N}\}$ |
| $\{ post: PriorityQueue\{\dots, (e_n, p_n), (e_{n+1}, p_{n+1})\} \}$ |

| $\boldsymbol{dequeue}$ ($\boldsymbol{PriorityQueue}$) |
|---|
| "$Removes\ the\ last\ element\ and\ its\ respective\ priority\ from\ the\ PriorityQueue.$" |
| $\{ pre: PriorityQueue \neq \{\ \ \} \}$ |
| $\{ post: PriorityQueue\{(e_1, p_1), \dots, (e_{n-1}, p_{n-1})\} \}\ returns\ and\ removes\ (e_n)$ |

| $\boldsymbol{incresePriority}$ ($\boldsymbol{PriorityQueue}$) |
|---|
| "$Increases\ the\ priorities\ of\ each\ PriorityQueue\ element.$" |
| $\{ pre: PriorityQueue \neq \{\ \ \} \}$ |
| $\{ post: PriorityQueue\{(e_1, p_1 + 1), \dots, (e_n, p_n + 1)\} \}$ |

|  |
|---|
| $\boldsymbol{prioritizeLowest}\ (\boldsymbol{PriorityQueue})$ |
| $"Decrements\ the\ priorities\ of\ each\ PriorityQueue\ element."$ |
| $\{\ pre: PriorityQueue\ \neq \{\quad\}\ \}$ |
| $\{\ post: PriorityQueue\{(e_1, p_1 - 1), \ldots, (e_n, p_n - 1)\}\ \}$ |

Analyzers:

|  |
|---|
| $\boldsymbol{peek}\ (\boldsymbol{PriorityQueue})$ |
| $"Returns\ the\ last\ value\ stored\ in\ the\ PriorityQueue."$ |
| $\{\ pre: PriorityQueue\ \neq \{\quad\}\ \}$ |
| $\{\ post: PriorityQueue\{(e_1, p_1), \ldots, (e_n, p_n)\}\ \}\ return\ (e_n)$ |

|  |
|---|
| $\boldsymbol{isEmpty}\ (\boldsymbol{PriorityQueue})$ |
| $"Check\ if\ the\ PriorityQueue\ is\ empty\ or\ not."$ |
| $\{\ pre: PriorityQueue <> \vee\ PriorityQueue < \cdots, elem_n >\}$ |
| $\{\ post: PriorityQueue = \{\quad\} \rightarrow true\ \}\ otherwise\ false.$ |

|  |
|---|
| $\boldsymbol{size}\ (\boldsymbol{PriorityQueue})$ |
| $"Returns\ the\ number\ of\ elements\ in\ the\ PriorityQueue."$ |
| $\{\ pre: PriorityQueue <> \vee\ PriorityQueue < \cdots, elem_n >\}$ |
| $\{\ post: return\ n\ \}$ |

|  |
|---|
| **Queue ADT** |
| Queue $= \{< e_1, e_2, e_3, \ldots, e_n >$ ^ n $\geq$ 0 ^ first ^ last} |
| { $inv$: n ≥ 0 ^ queue.size = n ^ first = $e_1$ ^ last = $e_n$} |
| $Operations$:<br><br>— $Queue$: → $Queue$<br>— $enqueue: Queue\ x\ element$ → $Queue$<br>— $dequeue: Queue$ → $element$<br>— $peek: Queue$ → $element$<br>— $isEmpty: Queue$ → $boolean$<br>— $size: Queue$ → $Integer$ |

Constructor:

| $\boldsymbol{Queue}$ () |
|---|
| "$Generates$ an empty queue." |
| $\{\, pre: true \,\}$ |
| $\{\, post: Queue <>\}$ |

Modifiers:

| $\boldsymbol{enqueue}$ $(\boldsymbol{Queue}, \boldsymbol{element})$ |
|---|
| "$Adds$ a new element at the last position in the queue." |
| $\{\, pre: queue = < e_1, e_2, e_3, \ldots, e_n > or\ queue = <>\}$ |
| $\{\, post: queue = < e_1, e_2, e_3, \ldots, e_n, element > or\ queue = < element >\}$ |

| $\boldsymbol{dequeue}$ $(\boldsymbol{Queue})$ |
|---|
| "Removes the first element from the queue and show the deleted element." |
| $\{\, pre: queue = < e_1, e_2, e_3, \ldots, e_n > or\ queue \ \neq <>\}$ |
| $\{\, post: queue = < e_2, e_3, \ldots, e_n >\}\ returns\ and\ removes\ (e_1)$ |

Analyzers:

| $\boldsymbol{peek}$ $(\boldsymbol{Queue})$ |
|---|
| "$Returns$ the value of the first element in the queue." |
| $\{\, pre: Queue \ \neq \{\quad\}\,\}$ |
| $\{\, post: element\ e_1 \,\}$ |

| $\boldsymbol{isEmpty}$ $(\boldsymbol{Queue})$ |
|---|
| "$Check\ if\ the\ Queue\ is\ empty\ or\ not.$" |
| $\{\, pre: Queue <> \vee Queue < \cdots, elem_n >\}$ |
| $\{\, post: Queue = \{\quad\} \rightarrow true \,\}\ otherwise\ false.$ |

| $\boldsymbol{size}$ $(\boldsymbol{Queue})$ |
|---|
| "$Returns\ the\ number\ of\ elements\ in\ the\ Queue.$" |
| $\{\, pre: Queue <> \vee Queue < \cdots, elem_n >\}$ |
| $\{\, post: Positive\ Integer \,\}$ |