|  |
| --- |
| **TAD Hash Table** |
| HT = {size=<size>, table=<table>}  table =<Node1, Node2, Node3, Noden>  Node = <K, V, next> |
| {inv: HT.table.length = HT.size ^ ∀ x, y ∈ HT.table, x ≠ y ⇒ hash(x) ≠ hash(y) } |
| Constructor operations:   * HT: <size> -> HT   Analysis operations:   * getValue HT x K - -> V * hash K -> Integer   Modification operations:   * add HT x K x V -> String * remove HT x K - -> String |

|  |
| --- |
| **HashTable()**  "Creates a new HashTable"  { pre: TRUE }  { post: HashTable = {table = <table>} } |

|  |
| --- |
| **add(HashTable, K, V)**  "Adds a new node to the hash table"  { pre: BST = { Root <root>}, … }  {post: If there is no collision, a new node is added to the table. If there is a collision, the new node is added at the end. Return a message with the operation result } |

|  |
| --- |
| **getValue(HashTable, K)**  "Returns the value associated with the given key"  { pre: TRUE }  { post: Returns the value associated with the given key or null if it is not found } |

|  |
| --- |
| **remove(HashTable, K)**  "Removes the node associated with the given key from the hash table"  { pre: TRUE }  { post: Returns a message indicating if the node was found and removed or if it was not found } |

|  |
| --- |
| **hash(K)**  "Calculates the index of the key in the hash table"  { pre: key is not null }  { post: non-negative integer less than the size of the hash table } |

|  |
| --- |
| **TAD Priority Queue** |
| PQ = {Size = <size>, Element = <element>, Comparator = <comparator> |
| {inv: PQ.Element.length == PQ.Size} |
| Constructor operations:   * PQ: -> PQ   Analysis operations:   * peek PQ -> V * isEmpty PQ -> Boolean   Modification operations:   * offer PQ x V x P -> String * poll PQ -> V * increasePriority PQ x V x P -> String |

|  |
| --- |
| **PriorityQueue()**  "Creates a new HashTable"  { pre: TRUE }  { post: HashTable = {table = <table>} } |