TAD TABLES

By: Juan Felipe Jojoa Crespo A00382042

Felipe Rojas Prado A00383819

Datastructures:

* HashTable
* HeapSort
* Priority Queue

TAD <HashTable>

HashTable = {SIZE, hashing, keyValidation, Htable}

Inv: {key, equalsIgnoreCase, Htable(allKeys)}

Primitive operations (Methods):

beforeHeapify(): -> data

isEmpty(): -> Boolean

isClear(): -> void

hashTableInsertion(size, hashing, keyValidation): -> new HashTable

setData(key, value): (new Hashtable \* key ^ value) -> void

getData(key): new HashTable -> value

removeData(key): (new HashTable \* key) -> boolean

TAD <Priority Queue>

priorityQueue = {SIZE, comparatorFunction (Interface comparable)}

inv: {comparatorFunction(keyA,keyB) = True}

Primitive Operations (Methods):

isClear(): -> void

sizeValidator(): -> data

peekEnd(): -> getNode

priorityQueueInsertion(SIZE): -> priorityQueue

removeFromQueue(): priorityQueue -> getNode

addQueue(data): priorityQueue \* getNode -> void

TAD <HeapSort>

HeapSort = {ancestor,rightNode, leftNode, SIZE}

Inv: {ancestor > rightNode & ancestor > leftNode}

Primitive Operations (Methods):

isClear(): -> void

isEmpty(): -> boolean

heapSortInsertion(SIZE): -> HeapSort

addHeap(data): HeapSort \* Node(Verifiy right and left validation) -> void

searchHeap(keyHeap): -> Node

removeHeap(keyHeap): HeapSort -> Node(Verifiy right and left validation and balance)