|  |
| --- |
| TAD Max priority queue |
| Max priority queue={size,comparator} |
| Inv: {comparator(a,b)= True for all elements 'a' and 'b' in the queue} |
| CreatePriorityQueue(size) -> MaxPriorityQueue  insert(TaskReminder task) -> void  increaseKey(int i, TaskReminder key) -> void  extractMaximum() -> TaskReminder  maximum() -> TaskReminder |

|  |
| --- |
| TAD Heap |
| Heap={parent,left,right,size} |
| Inv: {(parent>left ^ parent>right) V (parent<left ^ parent<right) |
| CreateHeap(size) -> Heap  insert(TaskReminder task) -> void  increaseKey(int i, TaskReminder key) -> void  extractMaximum() -> TaskReminder  maximum() -> TaskReminder  getHeapSize() -> int  setHeapSize(int heapSize) -> void  isEmpty() -> boolean  maxHeapify(int i) -> void |

|  |
| --- |
| TAD Hash table |
| HashTable = <size, hashFunction, keyEqualityFunction, table> |
| Inv: { key ≠ table(keys)} |
| Primitive Operations:  CreateHashTable(size, hashFunction, keyEqualityFunction) -> HashTable  insert(K key, V value) -> void  search(K key) -> Node<K, V>  delete(K key) -> void  hash(K key) -> int  isEmpty() -> boolean  getTable() -> Node<K, V>[] |

|  |
| --- |
| TAD <Queue > |
| Queue={add,poll,size} |
| Inv: {comparator(a,b)= True ^ (Q={x1,x2,x3,x4,x5…xn} ^ Q.poll()=x1)} |
| Primitive Operations:  CreateQueue() -> Queue  enqueue(K key, V value) -> void  dequeue() -> V  peek() -> Node<K, V>  isEmpty() -> boolean  size() -> int  getHead() -> Node<K, V>  setHead(Node<K, V> head) -> void |

**Max Priority Queque**

|  |
| --- |
| Createpriorityqueue(Size)  “Creates a new priority queque”  {pre: Size}  {pos: print priorityqueque data}  CreatePriorityQueue(size) -> MaxPriorityQueue  Enqueue(data) -> void  Dequeue() -> data  Peek() -> data  Size() -> data  IsEmpty() -> boolean |

|  |
| --- |
| enqueue(data)  “Adds data to queque”  {pre: data}  {pos: print priorityqueque data with enquque data } |

|  |
| --- |
| Dequeue()  “Removes data from queque ”  {pre: !=null}  {pos: print priorityqueque data without Dequeque data } |

|  |
| --- |
| peek ()  “print queque ”  {pre: !=null}  {pos: prints queque} |

|  |
| --- |
| size()  “Determines queque size”  {pre: !=null}  {pos: queque size} |

**Heap**

|  |
| --- |
| CreateHeap(size)  “Creates heap”  {pre: size }  {pos: prints created heap} |

|  |
| --- |
| InsertHeap(data)  “Inserts data in heap”  {pre: data}  {pos: prints Heap} |

|  |
| --- |
| DeleteHeap(data)  “Deletes data in heap”  {pre: data y !=null}  {pos: prints Heap } |

|  |
| --- |
| Find(data)  “finds element in heap”  {pre: data y !=null}  {pos: prints element in heap} |

|  |
| --- |
| sizeHeap()  “calculates size of heap”  {pre: !=null}  {pos: } |

|  |
| --- |
| isEmpty()  “Determine if the heap is empty”  {pre: True}  {pos: boolean} |

|  |
| --- |
| Heapify()  “Guarantees the order property of the heap”  {pre: !=null}  {pos: heap ordered} |

|  |
| --- |
| MaxHeapify()  “Guarantees the property of max heap, the first element is greater than all the elements”  {pre: !=null}  {pos: heap ordered} |

|  |
| --- |
| MinHeapify()  “Guarantees the property of min heap, the first element is smaller than all the elements”  {pre: !=null}  {pos: heap ordered} |

**HashTable**

|  |
| --- |
| createqueue(size, hashFunction, keyEqualityFunction)  “Creates a new hash table”  {pre: size, hashFunction, keyEqualityFunction }  {pos: String “hashTable created”} |

|  |
| --- |
| put(key, value)  “places value in a key”  {pre: key, value y hashtable!=null}  {pos: String “placed”} |

|  |
| --- |
| Get(key)  “gets value from key”  {pre: key }  {pos: value |

|  |
| --- |
| Remove(key)  “removes value from key”  {pre: key y !=null }  {pos: prints “removed: ” + value} |

|  |
| --- |
| ContainsKey(key)  “finds if it has a key”  {pre: key y ¡=null}  {pos: boolean} |

**Queue:**

|  |
| --- |
| CreateHashTable(add,poll,size)  “Creates a new queue”  {pre add,poll,size }  {pos: String “queue created”} |

|  |
| --- |
| add(element)  “adds a element to the queue”  {pre: element!=null y queue !=null}  {pos: String “added”} |

|  |
| --- |
| poll ()  “takes the first element out of the queue”  {pre: queue ¡= null}  {pos: Element} |

|  |
| --- |
| size ()  “find and return the queue size”  {pre: queue !=null }  {pos: prints size} |

|  |
| --- |
| isEmpty()  “determines if the queue is empty”  {pre: queue! = null}  {pos: boolean} |