**ADT Design**

* Daniel José Plazas (A00400085)
* Sebastián Erazo (A00400086)
* Juan Manuel Casanova (A00400090)

|  |
| --- |
| **Hash Table ADT** |
| HashTable = {table,PREDIFINED\_SIZE,size} |
| PREDIFINED\_SIZE=n, n^ size=size(table) ^ table[0,…,PREDIFINED\_SIZE-1] |
| **Primitive Operations:**  CreateHashTable : --> HashTable  IsEmpty : HashTable --> Boolean  Size : HashTable --> Integer |

|  |
| --- |
| **CreateHashTable** \_ --> HashTable |
| \*Creates an empty HashTable\* |
| {pre: PREDIFINED\_SIZE>0} |
| {pos: size=0, table={}} |

**-------------------------------------------------------------------------------------------------------------------------**

|  |
| --- |
| **Stack ADT** |
| Stack = {<e1,e2,e3...,en>,top} |
| 0<=n ^ size(Stack)=n ^ top = en |
| **Primitive Operations:**  CreateStack : --> Stack  Push : StackXElement --> Stack  Peek : Stack --> Element  Pop : Stack --> Element  IsEmpty : Stack --> Boolean  Size : Stack --> Integer |

|  |
| --- |
| **CreateStack** \_ --> Stack |
| \*Creates an empty Stack\* |
| {pre: TRUE} |
| {pos: Stack={,NILL}} |

|  |
| --- |
| **Push** StackXElement --> Stack |
| \*Adds an element to the stack\* |
| {pre: Stack={<e1,e2,…,en>,en} and Element, or Stack={,NILL} and Element} |
| {pos: Stack={<e1,e2,…,en,Element>,Element} or Stack{<Element>,Element}} |

|  |
| --- |
| **Peek** Stack-->Element |
| \*Gets and returns the top element in the Stack\* |
| {pre: Stack{,NILL}, for example Stack={<e1,e2,e3,e4,…,en>,en}} |
| {pos: en} |

|  |
| --- |
| **Pop** Stack --> Element |
| \*Deletes and return the top element in the Stack\* |
| {pre: Stack{,NILL}, for example Stack={<e1,e2,…,en>,en}} |
| {pos: en and Stack={<e1,e2,…,en-1>,en-1}} |

|  |
| --- |
| **IsEmpty** Stack --> Boolean |
| \*Verifies if the Stack is Empty\* |
| {pre: Stack={,NILL} or Stack{,NILL}, for example Stack={<e1,e2,…,en>,en}} |
| {pos: FALSE or TRUE} |

|  |
| --- |
| **Size** Stack --> Integer |
| \*Counts and returns the number of elements in the Stack\* |
| {pre: Stack={,NILL} or Stack{,NILL}, for example Stack={<e1,e2,…,en>,en}} |
| {pos: 0 or n, n} |

**-------------------------------------------------------------------------------------------------------------------------**

|  |
| --- |
| **Queue ADT** |
| Queue = {<e1,e2,e3,…,en>,top,bottom} |
| 0<=n ^ size(Queue)=n ^ top=e1 ^ bottom=en |
| **Primitive Operations:**  CreateQueue : --> Queue  IsEmpty : Queue --> Boolean  Peek : Queue --> Element  Add : QueueXElement --> Queue  Poll : Queue --> Element  Size : Queue --> Integer |

|  |
| --- |
| **CreateQueue** \_ --> Queue |
| \*Creates an Empty Queue\* |
| {pre: TRUE} |
| {pos: Queue={,NILL,NILL}} |

|  |
| --- |
| **IsEmpty** Queue --> Boolean |
| \*Verifies if the Queue is Empty\* |
| {pre: Queue={,NILL,NILL} or Queue{,NILL,NILL}, for example Queue={<e1,e2,e3,…,en>,e1,en}} |
| {pos: FALSE or TRUE} |

|  |
| --- |
| **Peek** Queue --> Element |
| \*Gets and returns the top element in the Queue\* |
| {pre: Queue{,NILL,NILL}, for example Queue={<e1,e2,e3,e4,…,en>,e1,en}} |
| {pos: en} |

|  |
| --- |
| **Add** QueueXElement --> Queue |
| \*Adds an element to the stack\* |
| {pre: Queue={<e1,e2,…,en>,e1,en} and Element, or Queue={,NILL,NILL} and Element} |
| {pos: Queue={e1,e2,…,en,Element},e1,Element} or Queue{<Element>,Element,Element}} |

|  |
| --- |
| **Poll** Queue --> Element |
| \*Deletes the top of the Queue\* |
| {pre: Queue{,NILL,NILL}, for example Q={<e1,e2,e3,…,en>,e1,en}} |
| {pos: e1 and Queue={<e2,e3,…,en>,e2,en}} |

|  |
| --- |
| **Size** Queue --> Integer |
| \*Counts and returns the number of elements in the Queue\* |
| {pre: Queue={,NILL,NILL} or Queue{,NILL,NILL}, for example Queue={e1,e2,e3,…,en},e1,en} |
| {pos: 0 or n, n} |

**-------------------------------------------------------------------------------------------------------------------------**

|  |
| --- |
| **Priority Queue ADT** |
| PriorityQueue={<e1,e2,…,en>,top} |
| 0<=n ^ size(PriorityQueue)=n ^ top=e1 ^ e1>=e2 ^ e2>e2+1 ^ ... ^ en-1>en |
| **Primitive Operations:**  CreatePriorityQueue : --> PriorityQueue  Add : PriorityQueueXElement --> PriorityQueue  Remove : PriorityQueue --> Element  Get : PriorityQueue --> Element  IsEmpty : PriorityQueue --> Boolean  Size : PriorityQueue --> Integer |

|  |
| --- |
| **CreatePriorityQueue** \_ --> PriorityQueue |
| \*Creates an Empty Priority Queue\* |
| {pre:TRUE} |
| {pos: PriorityQueue={,NILL}} |

|  |
| --- |
| **Add** PriorityQueueXElement --> PriorityQueue |
| \*Adds an element to the Priority Queue\* |
| {pre: PriorityQueue={,NILL} and Element, or PriorityQueue{,NILL}, for example PriorityQueue={<e1,e2,…,en>,e1}, and Element} |
| {pos: PriorityQueue={<Element>,Element} or PriorityQueue={<Element,e1,e2,…,en>,Element} or PriorityQueue={<e1,Element,e2,…,en>,e1} or PriorityQueue={<e1,e2,Element,…,en>,e1} or … or PriorityQueue={<e1,e2,…,Element,en>,e1} or PriorityQueue={<e1,e2,…,en,Element>,e1}} |

|  |
| --- |
| **Remove** PriorityQueue --> Element |
| \*Remove the top of the Priority Queue\* |
| {pre: PriorityQueue{,NILL}, for example PriorityQueue={<e1,e2,…,en>,e1}} |
| {pos: PriorityQueue={<e2,…,en>,e2}} |

|  |
| --- |
| **Get** PriorityQueue --> Element |
| \*Gets and returns the top of the Priority Queue\* |
| {pre: PriorityQueue{,NILL}, for example PriorityQueue={<e1,e2,e3,…,en>,e1}} |
| {pos: e1} |

|  |
| --- |
| **IsEmpty** PriorityQueue --> Boolean |
| \*Verifies if the Priority Queue is Empty\* |
| {pre: PriorityQueue={,NILL} or PriorityQueue{,NILL}, for example PriorityQueue={<e1,e2,e3,…,en>,e1}} |
| {pos: FASLE or TRUE} |

|  |
| --- |
| **Size** PriorityQueue --> Integer |
| \*Counts and returns the number of elements in the PriorityQueue\* |
| {pre: PriorityQueue={,NILL} or PriorityQueue{,NILL}, for example PriorityQueue={e1,e2,e3,…,en},e1} |
| {pos: 0 or n, n} |