**Test cases**

**Stack:**

**Note:** isEmpty() and getSize() are tested along with peak() test cases

* **peak():**

| **peakTest1** | |
| --- | --- |
| **Scenario** | |
| Stack s=empty | |
| **Operation** | **Result** |
| s.peak  s.isEmpty  s.getSize | Exception message  true  0 |

| **peakTest2** | |
| --- | --- |
| **Scenario** | |
| Stack s= [s1,s2] | |
| **Operation** | **Result** |
| s.push(s3)  s.peak()  s.isEmpty  s.getSize | s3  false  3 |

| **peakTest3** | |
| --- | --- |
| Scenario | |
| Stack s= [s1,s2] | |
| **Operation** | **Result** |
| s.pop()  s.peak  s.isEmpty  s.getSize | s1  False  1 |

* **pop():**

| **popTest1** | |
| --- | --- |
| **Scenario** | |
| Stack s= empty | |
| **Operation** | **Result** |
| s.pop() | Exception message |

| **popTest2** | |
| --- | --- |
| **Scenario** | |
| Stack s= [s1] | |
| **Operation** | **Result** |
| s.pop() | s1  Stack s= empty |

| **popTest2** | |
| --- | --- |
| **Scenario** | |
| Stack s= [s1,s2] | |
| **Operation** | **Result** |
| s.pop() | s2  Stack s= [s1] |

* **push(e):**

| **pushTest1** | |
| --- | --- |
| **Scenario** | |
| Stack s= [] | |
| **Operation** | **Result** |
| s.push(s1) | Stack s= [s1]  Stack s!= [] |

| **pushTest2** | |
| --- | --- |
| **Scenario** | |
| Stack s= [s1] | |
| **Operation** | **Result** |
| s.push(s2) | Stack s= [s1,s2] |

| **pushTest3** | |
| --- | --- |
| **Scenario** | |
| Stack s= [s1] | |
| **Operation** | **Result** |
| s.push(s1) | Stack s= [s1,s1] |

**HashTable:**

**Note:** hashFunction(k) is tested along with the other test cases.

* **chainedHashSearch(k):**

| **searchTest1** | |
| --- | --- |
| **Scenario** | |
| HashTable h= [h1] | |
| **Operation** | **Result** |
| h.chainedHashSearch(h1.getKey): | h1 |

| **searchTest2** | |
| --- | --- |
| **Scenario** | |
| HashTable h= [h1,h2] | |
| **Operation** | **Result** |
| h.chainedHashSearch(h1.getKey): | ;h1 |

| **searchTest3** | |
| --- | --- |
| **Scenario** | |
| HashTable h= [h1,h2] | |
| **Operation** | **Result** |
| h.chainedHashSearch(h3.getKey): | ;null |

* **chainedHashDelete(k):**

| **DeleteTest1** | |
| --- | --- |
| **Scenario** | |
| HashTable h= [h1] | |
| **Operation** | **Result** |
| h.chainedHashDelete(h1.getKey): | h1  HashTable h= empty |

| **DeleteTest2** | |
| --- | --- |
| **Scenario** | |
| HashTable h= [h1,h2] | |
| **Operation** | **Result** |
| h.chainedHashDelete(h1.getKey): | h1  HashTable h= [h2] |

| **DeleteTest3** | |
| --- | --- |
| **Scenario** | |
| HashTable h= [h1,h2] | |
| **Operation** | **Result** |
| h.chainedHashDelete(h3.getKey): | null |

* **chainedHashInsert(e):**

| **insertTest2** | |
| --- | --- |
| **Scenario** | |
| HashTable h= empty | |
| **Operation** | **Result** |
| h.chainedHashInsert(h1): | HashTable h= [h1] |

| **insertTest2** | |
| --- | --- |
| **Scenario** | |
| HashTable h=[h1] | |
| **Operation** | **Result** |
| h.chainedHashInsert(h2): | HashTable h= [h1,h2] |

| **insertTest3** | |
| --- | --- |
| **Scenario** | |
| HashTable h=[h1,h2] | |
| **Operation** | **Result** |
| h.chainedHashInsert(h3): | HashTable h= [h1,h2,,h3] |

**Heap:**

* **Heapify(int i) AND BuildHeap AND Insert (Insert always calls BuildHeap wich always calls Heapify)**

| **Insert-Build-HeapifyTest1 here we use Search Object** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.insert(**h1**)**  **heapPatients.insert(**h2**)** | Heap heapPatients = [h1,h2] No Changes |

| **Insert-Build-HeapifyTest2 here we use Search Object** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.insert(**h2**)**  **heapPatients.insert(**h1**)** | Heap heapPatients = [h1,h2,] Re organized since h1 has more priority |

| **Insert-Build-HeapifyTest3** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.insert(**h1 with some null value**)**  **heapPatients.insert(**null**)** | throw Exception() |

* **Maximun()**

| **MaximunTest1 here we use getHeapSize** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1h2] | |
| **Operation** | **Result** |
| **heapPatients.Maximum()** | print maximum (h1)  size still 2 |

| **MaximunTest2** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.Maximum()** | throw Exception |

**It’s only two cases since the third one should be heapPatients = null but when we create an object of the Controller class type or a Heap type their arrays get initialized so it’s impossible array = null**

* **ExtractMax()**

| **ExtractMax1 here we use getHeapSize** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1h2] | |
| **Operation** | **Result** |
| **heapPatients.ExtractMax()** | size of the heap has reduce in one unit  Heap heapPatients = [h2] |

| **ExtractMax2** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.ExtractMax()** | throw Exception |

**It’s only two cases since the third one should be heapPatients = null but when we create an object of the Controller class type or a Heap type their arrays get initialized so it’s impossible array = null**

* **IsEmpty()**

| **IsEmptyTest1** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.isEmpty()** | true |

| **IsEmptyTest2** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1,h2] | |
| **Operation** | **Result** |
| **heapPatients.isEmpty()** | false |

**It’s only two cases since the third one should be heapPatients = null but when we create an object of the Controller class type or a Heap type their arrays get initialized so it’s impossible array = null**

* **DeleteExact(int priority, long id, int position)**

| **DeleteExact here we use search object** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1,h2] | |
| **Operation** | **Result** |
| **heapPatients.DeleteExact(1, 112453023,1)** | Heap heapPatients = [h1] |

| **DeleteExact here we use search object** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.DeleteExact(1, 112453023,1)** | throw Exception |

**It’s only two cases since the third one should be heapPatients = null but when we create an object of the Controller class type or a Heap type their arrays get initialized so it’s impossible array = null**

* **IncreaseKey(int oldPriority, long id, int newPriority)**

| **IncreaseKey1 here we use search object** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1,h2] | |
| **Operation** | **Result** |
| **heapPatients.IncreaseKey(1, 112453023, 4) (Increase key of h2)** | Heap heapPatients = [h2,h1] |

| **IncreaseKey2** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.IncreaseKey(1, 112453023, 4)** | Throw Exception |

| **IncreaseKey3 here we use search object** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1,h2] | |
| **Operation** | **Result** |
| **heapPatients.IncreaseKey(2,11000204,1) (Decrease key of h1)** | Heap heapPatients = [h2,h1] |

* **getArray()**

| **getArrayTest1** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [] | |
| **Operation** | **Result** |
| **heapPatients.getArray()** | ArrayList<PatientNode> = [] |

| **getArrayTest2** | |
| --- | --- |
| **Scenario** | |
| Heap heapPatients = [h1,h2] | |
| **Operation** | **Result** |
| **heapPatients.getArray()** | ArrayList<PatientNode> = [h1,h2] |

**It’s only two cases since the third one should be heapPatients = null but when we create an object of the Controller class type or a Heap type their arrays get initialized so it’s impossible array = null**